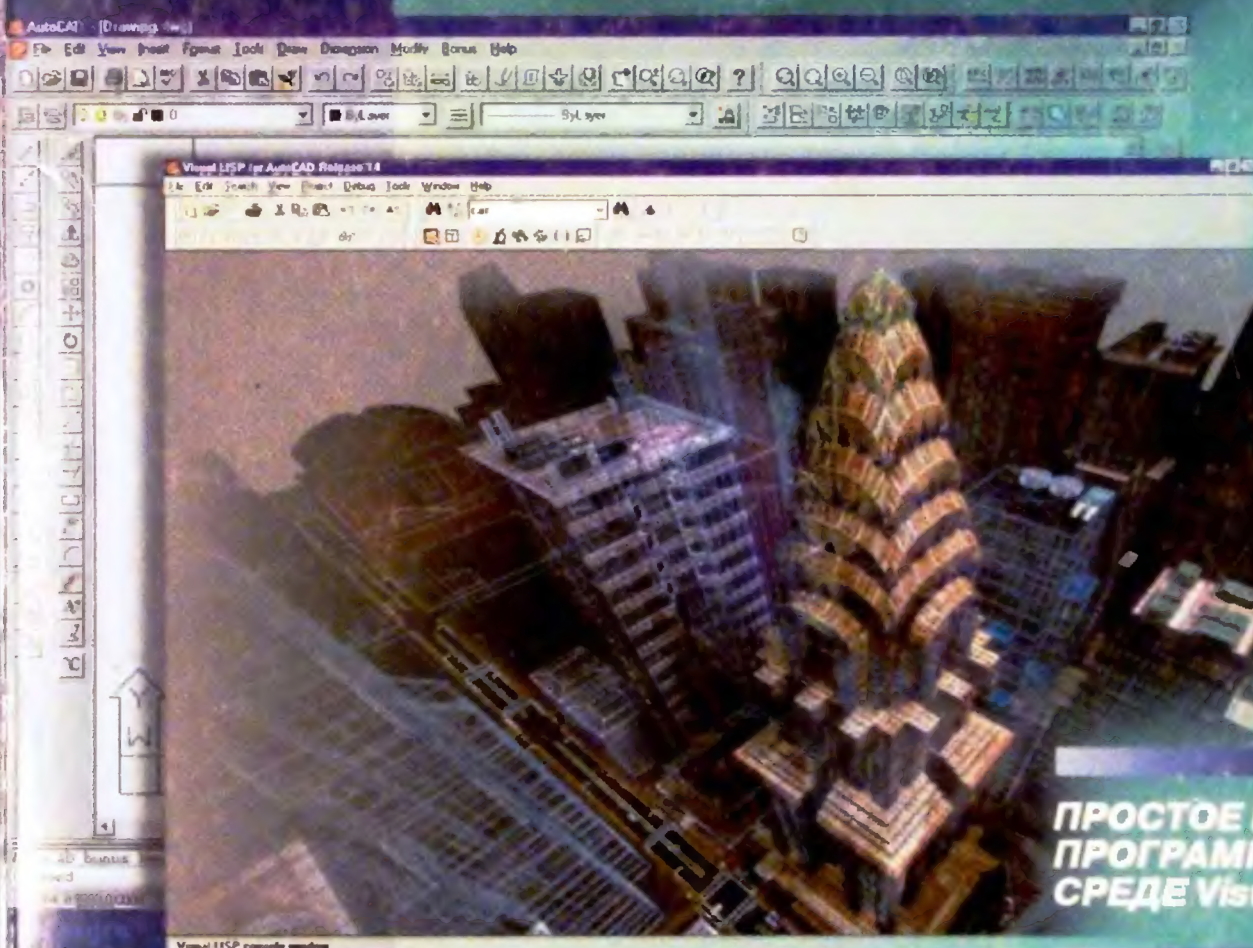


Кудрявцев Е.М.

AutoLISP

Программирование в AutoCAD 14



**ПРОСТОЕ И ЭФФЕКТИВНОЕ
ПРОГРАММИРОВАНИЕ В
СРЕДЕ Visual LISP**

**РАЗРАБОТКА
РЕЛЯЦИОННЫХ
БАЗ ДАННЫХ**

**СОЗДАНИЕ
ОБЪЕКТНО-
ОРИЕНТИРОВАННЫХ
СИСТЕМ**

**ВЫПОЛНЕНИЕ ЛЮБЫХ
ИНЖЕНЕРНЫХ РАСЧЕТОВ**



Для Windows 95/98/NT



Кудрявцев Е.М.

AutoLISP

Программирование в AutoCAD 14



Москва 1999

Кудрявцев Е.М.

**AutoLISP. Программирование в AutoCAD 14 / Кудрявцев Е.М. –
М.: «ДМК», 1999 – 368 с., ил.**

ISBN 5-89818-23-0

В настоящем издании рассматривается новая мощная интегрированная среда программирования Visual LISP для системы AutoCAD 14-й версии. Книга содержит описание структуры среды, главной и инструментальных панелей Visual LISP, текстового редактора, системы проверки синтаксиса, форматтера, компилятора, отладчика, встроенных функций и других элементов Visual LISP, а также правила ее запуска.

В отдельном разделе книги поясняются основные понятия и определения языка AutoLISP, рассказывается об этапах программирования, функциях, расширяющих возможности языка. Особое внимание уделяется вопросам разработки программ на языке AutoLISP и отладки в среде Visual LISP.

Автор приводит многочисленные примеры создания с использованием языка AutoLISP прототипов различных систем и подсистем для решения разнообразных задач.

Книга предназначена для широкого круга читателей: учащихся, студентов, инженеров, разработчиков автоматизированных систем конструирования и проектирования в самых разных областях деятельности.

ISBN 5-89818-23-0

© ДМК, Москва, 1999

Все права защищены. Любая часть этой книги не может быть воспроизведена в какой бы то ни было форме и какими бы то ни было средствами без письменного разрешения владельцев авторских прав.

Материал, изложенный в данной книге, многократно проверен. Но, поскольку вероятность технических ошибок все равно существует, издательство не может гарантировать абсолютную точность и правильность приводимых сведений. В связи с этим издательство не несет ответственности за возможные ошибки, связанные с использованием книги.

КРАТКОЕ СОДЕРЖАНИЕ

Глава 1

ИНТЕГРИРОВАННАЯ СРЕДА VISUAL LISP 13

Глава 2

ОСНОВЫ ПРОГРАММИРОВАНИЯ 47

Глава 3

**СОЗДАНИЕ ПАРАМЕТРИЧЕСКИХ ИЗОБРАЖЕНИЙ
ОБЪЕКТА 109**

Глава 4

**СОЗДАНИЕ ФРАГМЕНТОВ СИСТЕМ
ОБРАБОТКИ ИНФОРМАЦИИ 161**

Глава 5

**СОЗДАНИЕ ОБЪЕКТНО-ОРИЕНТИРОВАННЫХ
СИСТЕМ 191**

Глава 6

**СОЗДАНИЕ ПОДСИСТЕМ ДЛЯ ВЫПОЛНЕНИЯ
ИНЖЕНЕРНЫХ РАСЧЕТОВ 241**

Глава 7

**ОТЛАДКА И СОЗДАНИЕ ПРИЛОЖЕНИЙ В СРЕДЕ
VISUAL LISP 299**

СОДЕРЖАНИЕ

ПРЕДИСЛОВИЕ	11
--------------------------	----

Глава 1

ИНТЕГРИРОВАННАЯ СРЕДА VISUAL LISP	13
--	----

1.1. Структура и запуск Visual LISP	14
1.2. Главное меню Visual LISP	18
1.2.1. Падающее меню работы с файлами: File	21
1.2.2. Падающее меню редактирования: Edit	26
1.2.3. Падающее меню поиска: Search	28
1.2.4. Падающее меню просмотра: View	29
1.2.5. Падающее меню управления проектом: Project	30
1.2.6. Падающее меню отладки: Debug	30
1.2.7. Падающее меню средств управления системой: Tools	31
1.2.8. Падающее меню управления окнами: Window	32
1.2.9. Падающее меню помощи: Help	33
1.3. Панели инструментов Visual LISP	34
1.3.1. Панель инструментов просмотра: View	36
1.3.2. Панель инструментов: Tools	36
1.3.3. Панель инструментов отладки: Debug	37
1.3.4. Панель инструментов поиска: Search	38
1.4. Текстовый редактор Visual LISP	39
1.5. Компилирование программ и выход из Visual LISP	43

Глава 2

ОСНОВЫ ПРОГРАММИРОВАНИЯ	47
--------------------------------------	----

2.1. Основные понятия и определения	48
2.2. Основные этапы программирования на AutoLISP	52
2.3. Встроенные функции языка AutoLISP	58
2.3.1. Функции для ввода данных различного типа	59
2.3.2. Функции для манипулирования с данными	61
2.3.3. Функции для работы с числовыми данными и выражениями	63

2.3.4. Функции управления процессом вычисления (выполнения) функций	65
2.3.5. Функции проверки выполнения условий	67
2.3.6. Функции для вывода данных различного типа	68
2.3.7. Функции доступа к примитивам и средствам AutoCAD	69
2.3.8. Особые функции	71
2.4. Дополнительные функции языка AutoLISP	72
2.4.1. Примеры определения пользовательских функций	76
2.5. Программирование на AutoLISP в среде Visual LISP	82

Глава 3

СОЗДАНИЕ ПАРАМЕТРИЧЕСКИХ ИЗОБРАЖЕНИЙ ОБЪЕКТА

109

3.1. Параметрическое изображение плана объекта	110
3.1.1. Постановка задачи	110
3.1.2. Выявление основных особенностей, взаимосвязей и количественных закономерностей	110
3.1.3. Разработка последовательности действий и комплекса функций для параметрического изображения плана офиса	111
3.1.4. Программа (функция) параметрического изображения плана офиса	114
3.2. Создание параметрического изображения многоступенчатых объектов	117
3.2.1. Постановка задачи	117
3.2.2. Выявление основных особенностей, взаимосвязей и количественных закономерностей	117
3.2.3. Разработка последовательности действий и комплекса функций для параметрического изображения многоступенчатого объекта	118
3.2.4. Программа (функция) параметрического изображения многоступенчатого объекта в диалоговом режиме	121
3.2.5. Программа (функция) параметрического изображения многоступенчатого объекта	123
3.3. Нанесение размеров на многоступенчатом объекте и ввод текста	127
3.3.1. Постановка задачи	127
3.3.2. Выявление основных особенностей, взаимосвязей и количественных закономерностей	128

3.3.3. Разработка последовательности действий и программы нанесения размеров на многоступенчатом объекте ...	129
3.3.4. Программа параметрического изображения и указания размеров многоступенчатого объекта	134
3.3.5. Программа параметрического ввода текста из файла на чертеж	136
3.4. Создание параметрических изображений проекций конструкций	138
3.4.1. Постановка задачи	138
3.4.2. Выявление основных особенностей, взаимосвязей и количественных закономерностей	139
3.4.3. Разработка последовательности действий и программы параметрического изображения фронтальной проекции фермы	139
3.4.4. Программа параметрического изображения фронтальной проекции фермы в диалоговом режиме	141
3.4.5. Программа параметрического изображения горизонтальной проекции фермы	143
3.4.6. Программа параметрического изображения изометрии фермы ...	144
3.5. Создание параметрического изображения общего вида объекта	146
3.5.1. Постановка задачи	146
3.5.2. Выявление основных особенностей, взаимосвязей и количественных закономерностей	146
3.5.3. Разработка последовательности действий и программы параметрического изображения коттеджа	147
3.5.4. Программа параметрического изображения коттеджа	149
3.6. Создание параметрического изображения общего вида машины	150
3.6.1. Постановка задачи	150
3.6.2. Выявление основных особенностей, взаимосвязей и количественных закономерностей	150
3.6.3. Разработка последовательности действий и программы параметрического изображения основных узлов машины	150
3.6.4. Программа параметрического изображения ходовой части экскаватора	151
3.6.5. Программа параметрического изображения поворотной платформы экскаватора	152
3.6.6. Программа параметрического изображения стрелы экскаватора	153

3.6.7. Программа параметрического изображения рукояти экскаватора	155
3.6.8. Программа параметрического изображения ковша экскаватора	156
3.6.9. Программа параметрического изображения гидроцилиндра	158
3.6.10. Программа параметрического изображения общего вида экскаватора	159

Глава 4

СОЗДАНИЕ ФРАГМЕНТОВ СИСТЕМ ОБРАБОТКИ ИНФОРМАЦИИ

161

4.1. Преобразование алгебраических выражений	162
4.1.1. Постановка задачи	162
4.1.2. Выявление основных особенностей, взаимосвязей и количественных закономерностей	162
4.1.3. Разработка алгоритмов и комплекса функций для преобразования аналитических выражений	163
4.1.4. Программа преобразования аналитических выражений	169
4.1.5. Разработка последовательности действий и программы вычисления производных алгебраических функций	174
4.2. Обработка статистических данных	176
4.2.1. Постановка задачи	176
4.2.2. Выявление основных особенностей, взаимосвязей и количественных закономерностей	176
4.2.3. Разработка последовательности действий обработки экспериментально-статистических данных	177
4.2.4. Разработка фрагментов программы для обработки экспериментально-статистических данных	179
4.2.5. Программа для обработки экспериментально-статистических данных	183
4.3. Разработка простейшей реляционной базы данных	185
4.3.1. Постановка задачи	185
4.3.2. Выявление основных особенностей, взаимосвязей и количественных закономерностей	186
4.3.3. Разработка последовательности действий и программы создания базы данных	186
4.3.4. Программа создания базы данных в диалоговом режиме	187

4.3.5. Программа поиска данных в базе данных в диалоговом режиме ..	189
4.3.6. Программа обновления значений в базе данных в диалоговом режиме	190

Глава 5

СОЗДАНИЕ ОБЪЕКТНО-ОРИЕНТИРОВАННЫХ СИСТЕМ

СИСТЕМ	191
5.1. Расчет электрических и технических систем	192
5.1.1. Постановка задачи	192
5.1.2. Выявление основных особенностей, взаимосвязей и количественных закономерностей	192
5.1.3. Разработка последовательности действий и комплекса функций параметрического изображения всех элементов схемы	192
5.1.4. Комплекс функций параметрического изображения всех элементов электрической схемы	194
5.1.5. Программа параметрического изображения электрической схемы	200
5.1.6. Основные этапы автоматизации расчета электрических схем переменного тока	202
5.1.7. Основные этапы создания объектно-ориентированной подсистемы для расчета собственной частоты механической системы	209
5.2. Сетевое планирование и управление проектами	215
5.2.1. Постановка задачи	215
5.2.2. Разработка последовательности действий и комплекса функций параметрического изображения элементов сетевого графика	216
5.2.3. Комплекс функций параметрического изображения сетевого графика в диалоговом режиме	218
5.2.4. Разработка комплекса функций создания изображения сетевого графика по результатам расчета его на языке Fortran	221
5.2.5. Комплекс функций создания изображения сетевого графика по результатам расчета его на языке Fortran	229
5.3. Разработка экспертной системы	233
5.3.1. Постановка задачи	235
5.3.2. Выявление основных особенностей, взаимосвязей и количественных закономерностей	235

5.3.3. Разработка последовательности действий и комплекса функций для создания экспертной системы	236
5.3.4. Комплекс функций для создания экспертной системы	239

Глава 6

СОЗДАНИЕ ПОДСИСТЕМ ДЛЯ ВЫПОЛНЕНИЯ ИНЖЕНЕРНЫХ РАСЧЕТОВ

6.1. Расчет и представление динамических параметров	242
6.1.1. Постановка задачи	242
6.1.2. Выявление основных особенностей, взаимосвязей и количественных закономерностей	243
6.1.3. Разработка последовательности действий и комплекса функций расчета и параметрического изображения агрегата и всех его характеристик	245
6.1.4. Функция расчета и параметрического изображения агрегата и всех его характеристик	248
6.2. Расчет и создание параметрического изображения нагрузок в фермах	252
6.2.1. Постановка задачи	254
6.2.2. Выявление основных особенностей, взаимосвязей и количественных закономерностей	255
6.2.3. Разработка последовательности действий для расчета и параметрического изображения линий влияния	255
6.2.4. Разработка комплекса функций для расчета и параметрического изображения линий влияния	262
6.2.5. Комплекс функций для расчета и параметрического изображения линий влияния стрелы	266
6.3. Расчет и параметрическое изображение различных способов выполнения производственного процесса	270
6.3.1. Постановка задачи	270
6.3.2. Выявление основных особенностей, взаимосвязей и количественных закономерностей	270
6.3.3. Разработка последовательности действий и комплекса функций расчета и параметрического изображения различных способов выполнения производственного процесса	274
6.3.4. Функции расчета и параметрического изображения различных способов выполнения производственного процесса	277

6.4. Оптимизация загрузки оборудования	288
6.4.1. Постановка задачи	288
6.4.2. Выявление основных особенностей, взаимосвязей и количественных закономерностей	288
6.4.3. Разработка последовательности действий и комплекса функций для расчета и изображения оптимальной загрузки двух станков	289
6.4.4. Функция для расчета и изображения процесса оптимальной загрузки двух станков	295

Глава 7

ОТЛАДКА И СОЗДАНИЕ ПРИЛОЖЕНИЙ В СРЕДЕ VISUAL LISP

299

7.1. Отладка программ в среде Visual LISP	300
7.2. Создание приложений в среде Visual LISP	321
7.3. Встроенные функции Visual LISP	332

ПРИЛОЖЕНИЕ

345

Графические примитивы AutoCAD и примеры их создания с использованием AutoLISP	346
Команды графического редактора AutoCAD для редактирования объектов	350
Представление данных в системе AutoCAD	352

АЛФАВИТНЫЙ УКАЗАТЕЛЬ

356

ПРЕДИСЛОВИЕ

AutoCAD – это мощная, самая распространенная у нас и за рубежом инженерная система автоматизации проектирования самых разнообразных объектов: от плана офиса до космических станций.

Постоянно развивающаяся система AutoCAD состоит из трех основных компонентов: графического редактора AutoCAD, языка программирования высокого уровня AutoLISP и инструментальных средств для создания графического интерфейса пользователя.

Использование в системе AutoCAD языка AutoLISP не только значительно ускоряет процесс разработки проектной документации, но и позволяет создавать новые команды графического редактора, специализированные меню в среде AutoCAD, осуществлять доступ к графической базе данных и модернизировать ее, разрабатывать функции для решения самых разнообразных задач и, кроме того, создавать эффективные системы и подсистемы, связанные с обработкой информации, представленной в виде символов и чисел. Язык LISP был создан в 1962 году Дж. Маккарти, профессором Стенфордского университета для эффективного решения задач искусственного интеллекта. В настоящее время существует огромное количество версий этого языка, однако предпочтение отдается версии под названием COMMON LISP. Именно эта версия была поддержана Лабораторией искусственного интеллекта Массачусетского технологического института (США), там же создана LISP машина и в качестве языка системного программирования использован COMMON LISP. По синтаксису и соглашениям AutoLISP наиболее близок к COMMON LISP.

Формы представления программы и обрабатываемых ею данных в LISP одинаковы. И то и другое представляется в виде списочной структуры. Программы могут обрабатывать, а также преобразовывать другие программы и даже самих себя, что позволяет эффективно использовать LISP для решения широкого круга задач.

В настоящее время на языке LISP реализованы многочисленные программные продукты: редакторы GNU EMACS для различных языков программирования на ЭВМ семейства VAX, математическое обеспечение системы инженерного проектирования AutoCAD, системы аналитических преобразований, объектно-ориентированные системы, большинство экспертных систем и другие.

Для 14-й версии AutoCAD создана мощная интегрированная среда программирования Visual LISP. Она включает следующие функциональные компоненты:

- текстовый редактор, ориентированный на синтаксис языка AutoLISP и использующий язык кодирования цветом (DCL), что значительно упрощает чтение и отладку программ на языке AutoLISP;
- консоль, куда выдаются сообщения о результатах загрузки программы, различные диагностические сообщения и т. д.;
- форматтер, который преобразует текст программы и придает ему легко читаемый структурированный вид;
- эмулятор AutoLISP, обеспечивающий наглядность процесса выполнения программы;
- программу проверки синтаксиса, распознавания неправильных конструкций AutoLISP;
- встроенную систему проверки, обеспечивающую удобный доступ к переменным и значениям выражений с целью просмотра структуры данных и их изменений;
- отладчик, обеспечивающий высокий уровень отладки программ;
- контекстно-зависимые справки для функций AutoLISP;
- систему управления проектом, которая упрощает работу с приложениями;
- упаковщик компилируемых файлов AutoLISP в ARX-модули.

Visual LISP полностью поддерживает интерфейс Windows.

По сравнению с традиционными системами современные среды программирования, какой является интегрированная среда Visual LISP, позволяют повысить производительность программирования в несколько раз.

Целью настоящей книги является описание не только новой мощной интегрированной среды программирования Visual LISP для системы AutoCAD 14-й версии, но и процессов создания различных систем с ее использованием.

Книга содержит большое количество рисунков, иллюстрирующих интегрированную среду Visual LISP, программирование на языке AutoLISP и решение конкретных задач.

Автор далек от мысли, что книга свободна от недостатков, и с благодарностью примет все замечания и предложения по ее улучшению, которые просит направлять по адресу: books@dmk.ru.

ИНТЕГРИРОВАННАЯ СРЕДА VISUAL LISP

Структура и запуск	
Visual LISP	14
Главное меню	
Visual LISP	18
Панели инструментов	
Visual LISP	34
Текстовый редактор	
Visual LISP	39
Компилирование программ	
и выход из Visual LISP	43

Эта глава посвящена описанию интегрированной среды программирования Visual LISP. Здесь рассматриваются структура среды, правила ее запуска, описываются главная и инструментальные панели, текстовый редактор и другие элементы Visual LISP.

1.1. Структура и запуск Visual LISP

Visual LISP (VLISP) – это интегрированная среда разработки программ на языке программирования AutoLISP в системе AutoCAD 14, которая значительно облегчает процесс создания программы, ее изменения, тестирования и отладки. Кроме того, Visual LISP обеспечивает эффективное использование автономных приложений OBJECTARX, написанных на языке AutoLISP.

Visual LISP включает следующие функциональные компоненты:

- текстовый редактор, ориентированный на синтаксис языка AutoLISP и использующий язык кодирования цветом (DCL), что значительно упрощает чтение и отладку программ на языке AutoLISP;
- консоль, куда выдаются сообщения о результатах загрузки программы, различные диагностические сообщения и т. д.;
- форматтер, который преобразует текст программы и придает ему легко читаемый структурированный вид;
- эмулятор AutoLISP, обеспечивающий наглядность процесса выполнения программы;
- программу проверки синтаксиса, распознавания неправильных конструкций AutoLISP;
- компилятор, который уточняет выполнение программы, ускоряет, а также обеспечивает безопасную и эффективную систему выдачи результатов;
- встроенную систему контроля, которая обеспечивает удобный доступ к переменным и значениям выражений с целью просмотра структуры данных и их изменений;
- отладчик, обеспечивающий высокий уровень отладки программ;
- контекстно-зависимые справки для функций AutoLISP;
- систему управления проектом, которая упрощает работу с приложениями;
- упаковщик компилируемых файлов AutoLISP в ARX-модули.

Visual LISP имеет собственный набор окон и меню, который отличается от соответствующего набора AutoCAD. Однако запуск интегрированной среды Visual LISP производится из системы AutoCAD 14-й версии. Вот почему при работе с Visual LISP систему AutoCAD необходимо хранить на рабочем столе в открытом виде. Если AutoCAD свернут, то для продолжения работы с Visual LISP окно AutoCAD необходимо восстановить и сделать его активным. При работе с графикой интегрированная среда Visual LISP взаимодействует с системой AutoCAD, отвечая на подсказки

относительно ввода. Запуск Visual LISP производится только после запуска системы AutoCAD. При этом возможны два варианта.

Первый из них предусматривает загрузку приложения Visual LISP с помощью меню. AutoCAD имеет иерархическую систему меню, состоящую из главного меню и системы падающих и всплывающих меню (подменю). Главное меню AutoCAD – это набор пунктов меню для вызова падающих меню. Набор пунктов главного меню располагается во второй строке рабочего стола AutoCAD 14-й версии. Падающее меню AutoCAD – это набор пунктов меню для вызова всплывающего меню, диалогового окна или команды AutoCAD. Набор пунктов падающего меню располагается под соответствующим пунктом главного меню. Всплывающее меню AutoCAD – это набор пунктов меню для вызова всплывающего меню, диалогового окна или команды AutoCAD. Набор пунктов всплывающего меню располагается справа от выбранного пункта меню. Выбор пункта меню можно выполнить щелчком по нему левой кнопкой мышки.

Чтобы произвести загрузку, вызовите в главном меню AutoCAD пункт **Tools** (Инструменты) и выберите в падающем меню пункт **Load Application...** (Загрузить приложение...). На экране появится диалоговое окно загрузки (рис. 1.1) **Загрузить AutoLISP, ADS, и ARX Файлы** (Load AutoLISP, ADS, and ARX Files).

Чтобы выбрать приложение Visual LISP, щелкните в диалоговом окне загрузки по кнопке **File** (Файл), и на экране появится диалоговое окно выбора файлов **Выберите файл AutoLISP, ADS или ARX** (Select AutoLISP, ADS, or ARX File) (рис. 1.2).

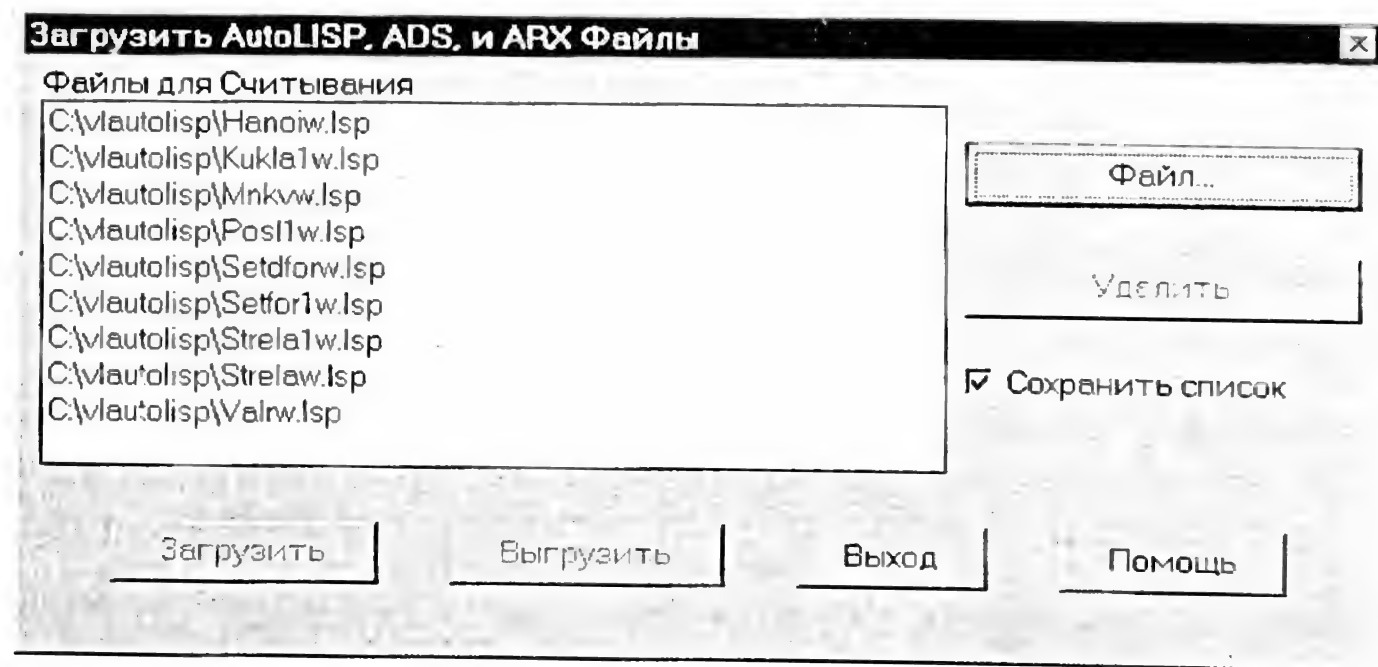


Рис. 1.1. Диалоговое окно загрузки файлов AutoLISP, ADS и ARX

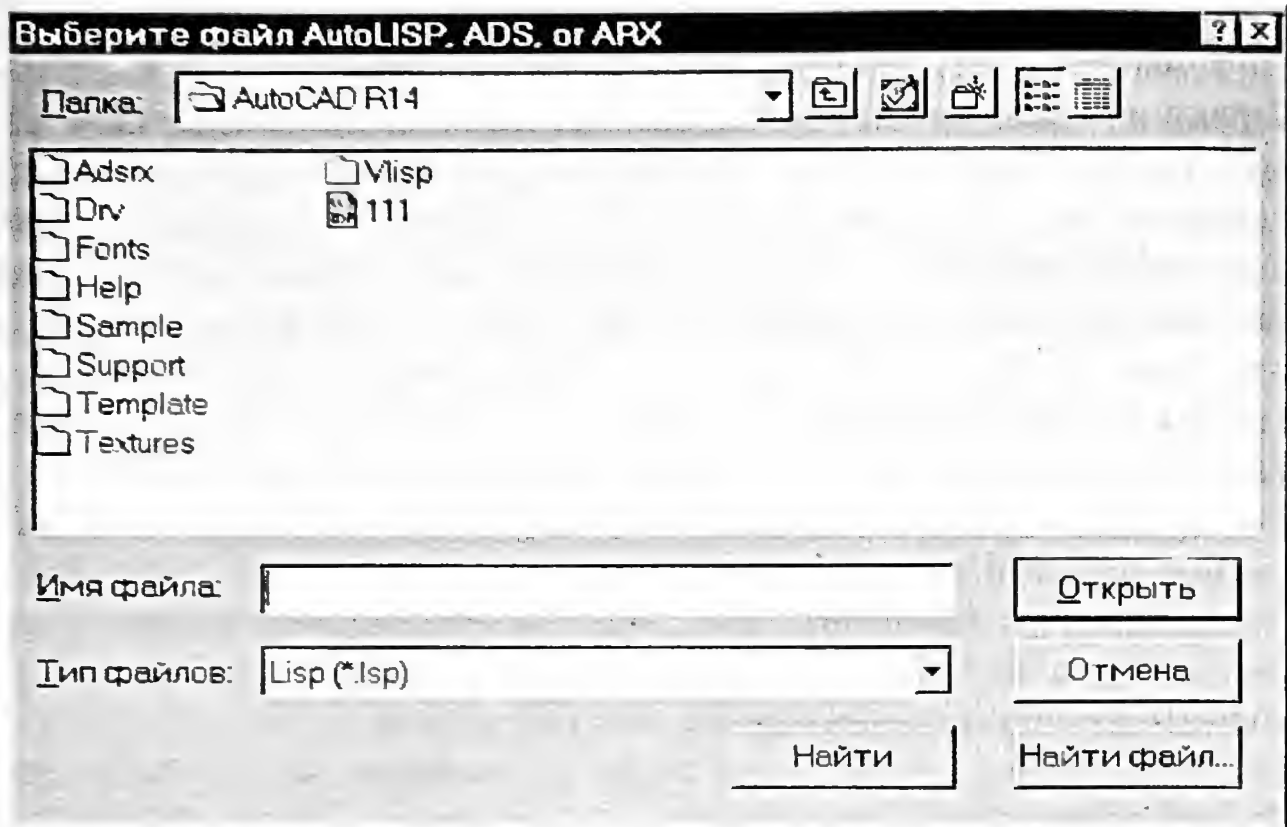


Рис. 1.2. Диалоговое окно выбора файлов AutoLISP, ADS или ARX

Найдите в этом окне папку (каталог) под названием Vlisp и дважды щелкните по ней мышкой. На экране появятся файлы папки (каталога) Visual LISP (рис. 1.3).

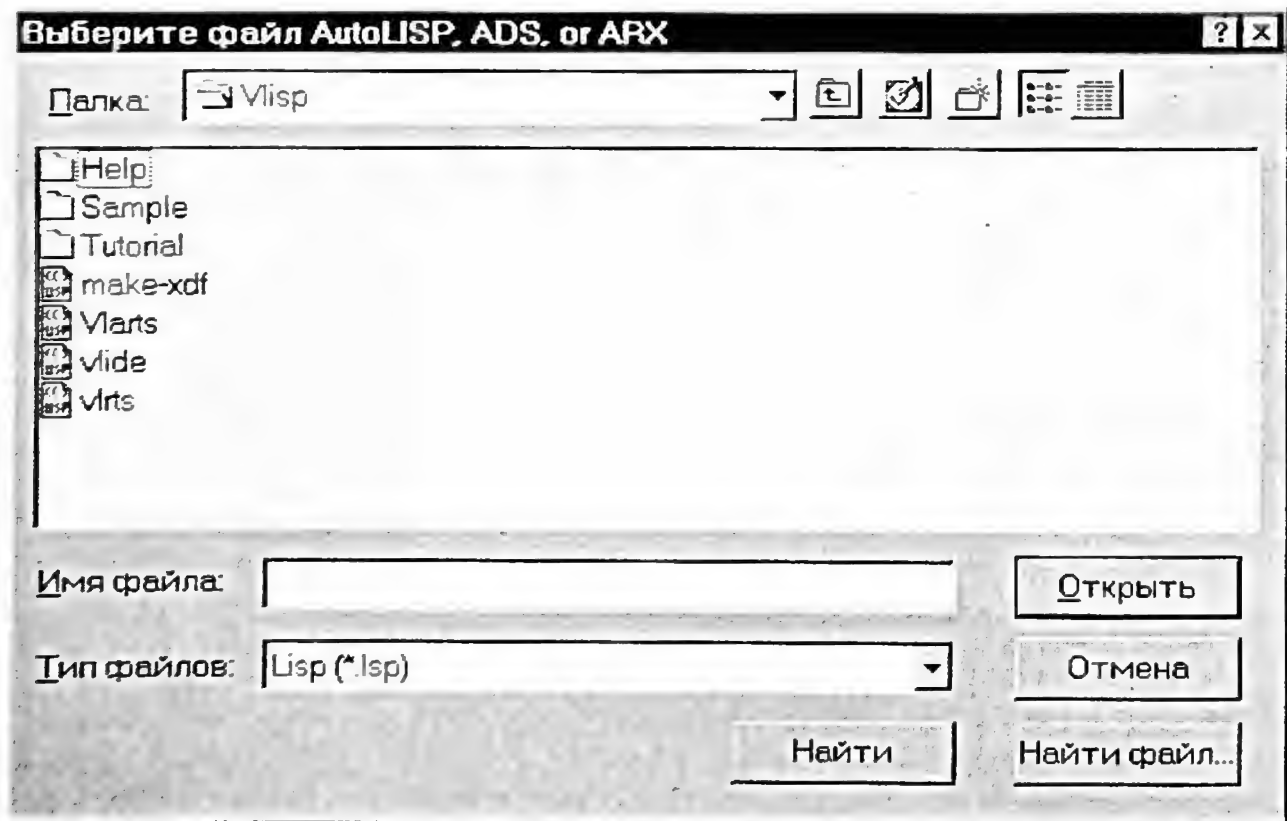


Рис. 1.3. Диалоговое окно выбора файлов AutoLISP, ADS или ARX

Диалоговое окно выбора файла может содержать по умолчанию только файлы с расширением LSP. В таком случае вам предстоит выбрать в поле **Тип файлов** (Files of type) расширение ARX. Для этого щелкните мышкой по расположенной рядом кнопке со стрелкой вниз. Откроется список расширений. Выберите расширение ARX и щелкните по нему мышкой. В том же диалоговом окне появится другой набор файлов. Выделите файл под именем Vlide и щелкните по кнопке **Открыть**. На экране вновь появится диалоговое окно загрузки **Загрузить AutoLISP, ADS, и ARX Файлы** (Load AutoLISP, ADS, and ARX Files).

Чтобы файл Vlide можно было использовать в дальнейшем, включите флажок **Сохранить список** (Save List). В квадратике рядом с надписью **Сохранить список** (см. рис. 1.1) появится галочка.

Щелкните по кнопке **Load** (Загрузить). На экране появится главное окно Visual LISP (рис. 1.4).

Второй вариант запуска предусматривает загрузку приложения Visual LISP с помощью команды VLIDE. В командной строке AutoCAD необходимо набрать команду VLIDE, нажать клавишу **Enter** (Ввести). На экране появится главное окно Visual LISP (рис. 1.4).

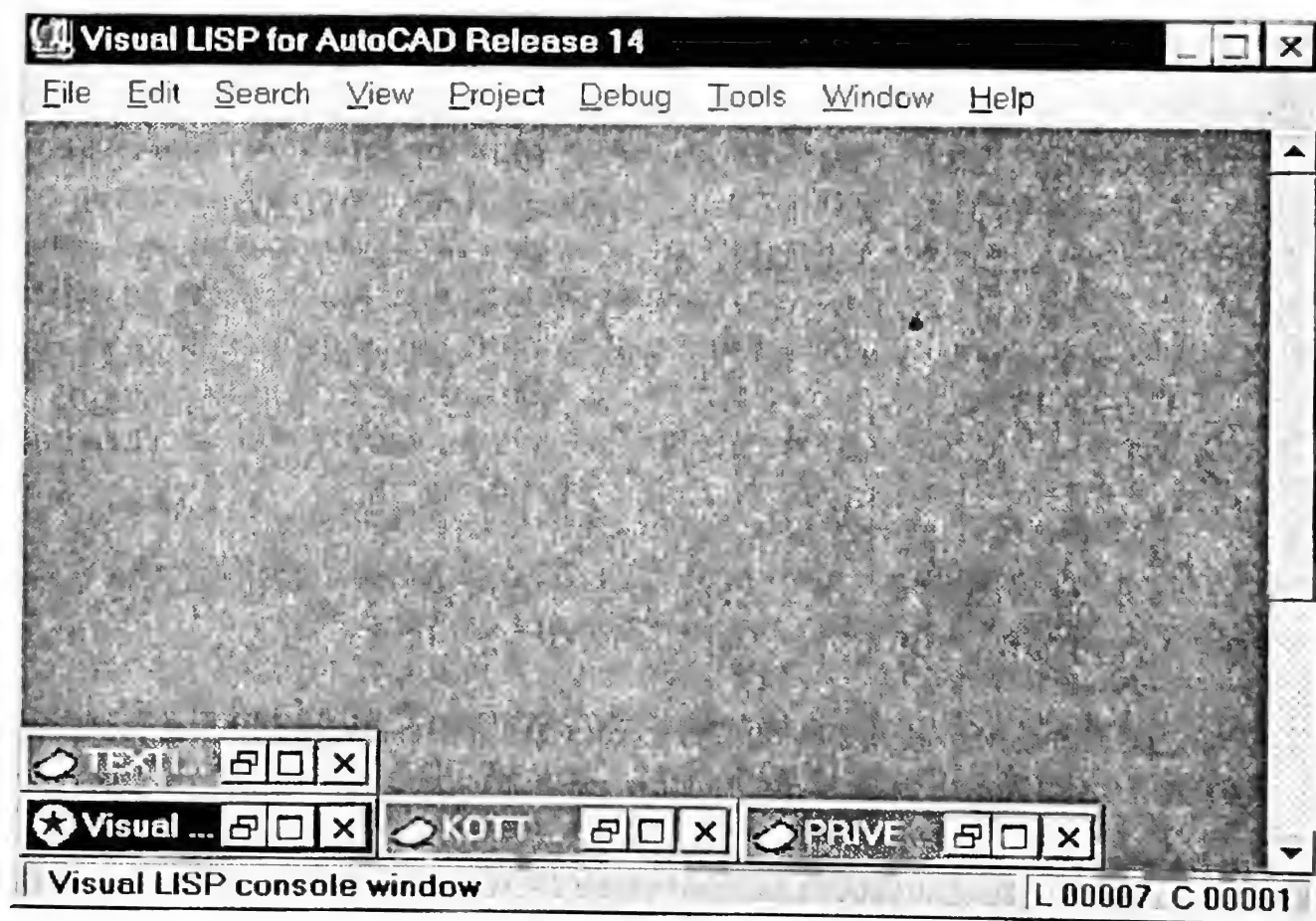


Рис. 1.4. Главное окно Visual LISP

В первой строке (строке заголовка) главного окна Visual LISP указано название окна **Visual LISP for AutoCAD Release 14** (Visual LISP для AutoCAD 14-й версии).

Во второй строке главного окна Visual LISP расположены пункты главного меню. Нижняя строка главного окна Visual LISP – строка состояния системы, в которой дается краткое описание функции команды.

1.2. Главное меню Visual LISP

Главное меню представляет собой систему, которая обеспечивает доступ ко всем средствам Visual LISP. По своей сути главное меню – основной управляющий центр интегрированной среды Visual LISP. Роль дополнительных центров играют панели инструментов и отдельные кнопки, за которыми закреплены команды, используемые наиболее часто.

Система меню Visual LISP – это хорошо скоординированная совокупность падающих и всплывающих меню. После щелчка мышкой по любому пункту главного меню или нажатия горячих клавиш (**Hot Keys**) на экране появляется соответствующее падающее меню. Горячая клавиша выделяется в названии пункта меню путем подчеркивания одной буквы. Для того чтобы с помощью клавиатуры получить быстрый доступ к пункту главного меню, а значит, и к соответствующему падающему меню, необходимо нажать клавишу **Alt** и, удерживая ее, нажать ту алфавитную клавишу, название которой подчеркнуто. Например, для быстрого обращения к пункту **View** (Просмотр) достаточно нажать комбинацию клавиш **Alt+V**. Чтобы вызвать соседнее падающее меню, необходимо нажать клавишу со стрелкой влево или вправо. Выделить любой пункт падающего или всплывающего меню (подменю) можно, щелкнув по нему мышкой или нажав клавишу со стрелкой вниз либо вверх. Для выполнения выбранной команды необходимо нажать клавишу **Enter** или горячую клавишу, название которой подчеркнуто. Чтобы закрыть падающее или всплывающее меню, достаточно нажать клавишу **Esc** или щелкнуть мышкой в поле окна вне меню.

Содержание падающих меню может изменяться в зависимости от используемого окна Visual LISP. Падающее меню состоит из пунктов, которые могут быть пунктами подменю, диалоговыми или недиалоговыми командами.

Если после названия пункта меню стоит многоточие, это означает, что перед вами диалоговая команда и при ее выборе появится диалоговое окно.

Когда в правой части пункта меню имеется треугольная стрелка, то при выборе данного пункта откроется подменю (всплывающее меню).

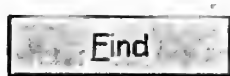
В случае, когда яркость пункта меню понижена, этот пункт не доступен для использования.

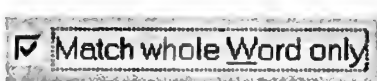
Если пункту меню предшествует флажок (галочка) или он появляется при выборе пункта, то это свидетельствует о том, что данный пункт может находиться во включенном или выключенном состоянии. При наличии флажка слева пункт меню считается включенным, при его отсутствии — выключенным.

Подчеркнутая буква в названии пункта меню обозначает горячую клавишу (Hot Key), которая может быть использована для быстрого доступа к команде или меню. Клавиши, указанные справа от пункта меню, являются клавишами-акселераторами (Shortcut Keys, Acceleration Keys) и предназначены для оперативного доступа к пункту меню или команде.


В процессе выбора пункта меню часто появляются диалоговые окна, в которых предлагается ввести недостающую информацию, уточнить режимы выполнения команды и/или выбрать один из нескольких вариантов.

Диалоговые окна имеют ряд элементов управления:

 **кнопку управления (command button)**, которая выполняет некоторое действие. Чтобы выполнить это действие, необходимо вначале выделить кнопку с помощью клавиши **Tab (Shift+Tab)**, а затем нажать **Enter** или щелкнуть по кнопке мышкой;

 **флажок (check box, option)**, который задает или уточняет режим выполнения команды. Он может находиться в двух состояниях — включенном (в квадратике слева стоит галочка) или выключенном (галочка отсутствует). Если флажок включен, то при выполнении команды будет осуществлено действие, указанное справа от квадратика. Смена состояния флажка производится щелчком мышки или путем выделения надписи справа с помощью клавиши **Tab (Shift+Tab)**, а затем нажатия клавиши **Space**;

 **переключатель (radio buttons)**, представляющий собой перечень взаимоисключаемых вариантов, из которых выбирается какой-либо один. Эти варианты называются положениями переключателя. Текущее положение отмечается точкой в кружочке слева. Для установки переключателя в нужное положение необходимо щелкнуть по этому кружочку мышкой или выделить надпись справа с помощью клавиши **Tab (Shift+Tab)**, а затем нажать клавишу **Space**;

 **поле ввода (text box)**, которое позволяет набирать, оставлять или редактировать

имеющуюся последовательность символов. Поле ввода предоставляет возможность вставлять текст из буфера обмена или копировать его туда. Чтобы обеспечить ввод символов в поле с клавиатуры, необходимо щелкнуть по полю ввода мышкой или ввести курсор в поле с помощью клавиши **Tab (Shift+Tab)**;

- **прокручиваемый список (scrolling list)**, который содержит те или иные элементы, в частности папки (каталоги), файлы, в определенной последовательности. Если элементов много, справа и/или снизу список имеет полосы прокрутки, которые дают возможность просмотреть весь список. Для выделения элемента из списка, щелкните по нему мышкой. Можно также использовать клавишу **Tab** и клавиши со стрелками;



раскрывающийся список (drop-down list), содержащий набор элементов. Из

этого набора выбирается единственный элемент, который оказывается видимым. Для выбора другого элемента нужно щелкнуть мышкой по кнопке в правой части списка. Когда список откроется, щелкните по выбранному элементу. Возможно, придется использовать приемы прокрутки. Как и в прокручиваемом списке, искать элемент можно по первой букве. Чтобы отменить действия по выделению другого элемента, щелкните мышкой вне зоны раскрывшегося списка или нажмите клавишу **Esc**.

Следует обратить внимание, что в диалоговых окнах выбора файлов (см. рис. 1.2), открытия файла **Open file to edit/view** (Открыть файл для редактирования или просмотра), сохранения файла **Save as**, загрузки файла **Load lisp file**, создания нового проекта **New project...** и других в правой части под строкой заголовка имеются следующие кнопки:



на один уровень вверх (Up One Level) – для перехода в каталог на один уровень выше;



обзор рабочего стола (View Desktop) – для показа программ на рабочем столе;



создание новой папки (New Folder) – для создания новой папки;



список (List) – для представления всех папок и файлов текущей папки;



таблица (Table) – для развернутого представления всех папок и файлов текущей папки (каталога) с указанием размера, типа и даты последнего изменения.

Далее рассмотрим падающие меню для каждого пункта главного меню.

<u>N</u> ew File	Ctrl-N
<u>O</u> pen File...	Ctrl-O
<u>R</u> eopen	
<u>S</u> ave	Ctrl-S
Save <u>A</u> s...	Ctrl-Alt-S
Save <u>A</u> ll	Alt-Shift-S
<u>C</u> lose	Ctrl-F4
Revert	
Clo <u>s</u> e All	
<u>P</u> rint...	Ctrl-P
Print Setup...	
<u>M</u> ake Application	Ctrl-Shift-M
Load <u>F</u> ile...	Ctrl-Shift-L
<u>E</u> xit	Alt-Q

<u>N</u> ew File	Ctrl-N
<u>O</u> pen File...	Ctrl-O
<u>R</u> eopen	
Save All	Alt-Shift-S
<u>P</u> rint...	Ctrl-P
Print Setup...	
<u>M</u> ake Application	Ctrl-Shift-M
Load <u>F</u> ile...	Ctrl-Shift-L
Toggle Console Log...	
<u>E</u> xit	Alt-Q

Рис. 1.5. Падающие меню работы с файлами (file) при активном окне текстового редактора (слева) и при активном окне консоли Visual LISP (справа)

1.2.1. Падающее меню работы с файлами: File

Выбор пункта **File** (Файл) главного меню или нажатие комбинации клавиш **Alt+F** вызывает падающее меню работы с файлами (рис. 1.5).

Падающее меню работы с файлами включает большой набор пунктов:

- **New File** (Новый файл) или нажатие комбинации клавиш **Ctrl+N** обеспечивает создание нового файла с помощью диалогового окна **<Untitled-0>** (Без названия-0) (рис. 1.6). Номер окна без названия определяется числом созданных, но еще не названных файлов. Нумерация еще не названных окон начинается с 0.

Если щелкнуть по значку, расположенному слева от названия окна, появится контекстное меню (рис. 1.6). Оно включает ряд пунктов:

- **Восстановить** (Restore) – восстанавливает размер окна до прежнего размера;
- **Переместить** (Move) – перемещает окно в другое место;
- **Размер** (Size) – изменяет размер окна. При выборе этого пункта указатель мышки изменит свою форму. Нажмите клавишу **↓**, **↑**, **←** или **→**, чтобы указать границу окна, которую вы намереваетесь переместить. Переместите границу, двигая мышку либо нажимая ту же или противоположную ей клавишу. Завершите операцию щелчком мышки или нажатием клавиши **Enter** (Ввод);

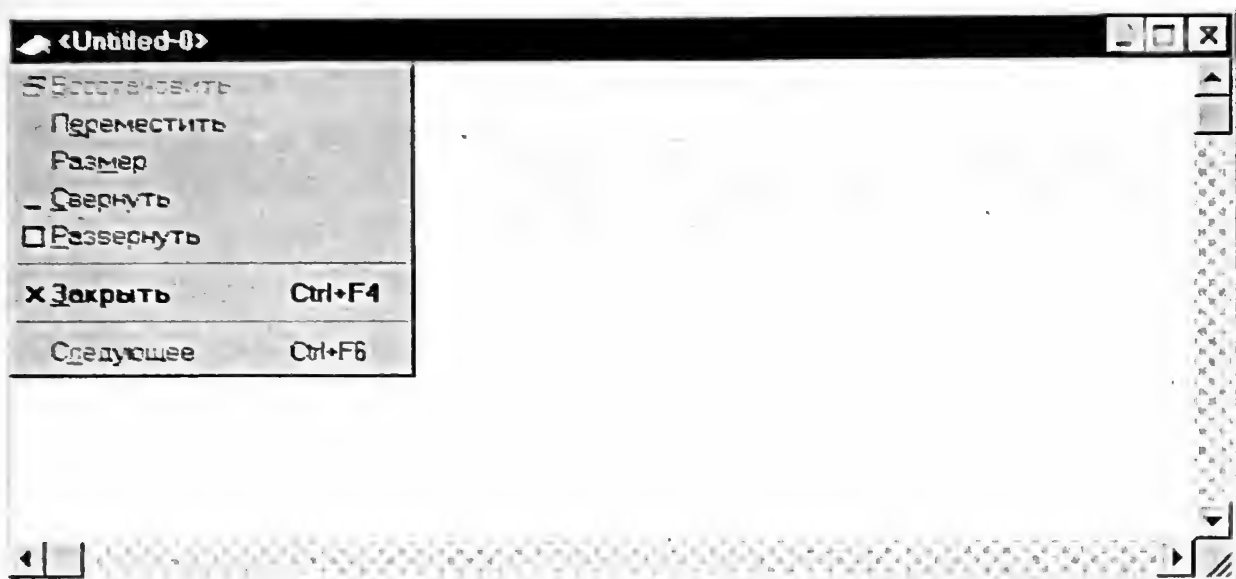


Рис. 1.6. Окно для создания нового файла

- **Свернуть (Minimize)** – сворачивает окно до минимальных размеров (уменьшенный размер заголовка окна);
- **Развернуть (Maximize)** – разворачивает нормальное или свернутое окно до максимально возможных размеров;
- **Заккрыть (Close)** – закрывает окно;
- **Следующее (Next)** – обеспечивает переход к следующему окну. Это же действие может быть выполнено путем нажатия комбинации клавиш **Ctrl+F6**.

В правой части строки заголовка текстового окна имеются три кнопки. Первая и вторая кнопки (Свернуть и Развернуть) позволяют изменять размер окна, правая (Заккрыть), закрывает окно;

- **Open File...** (Открыть файл) или нажатие комбинации клавиш **Ctrl+O** вызывает диалоговое окно открытия файла **Open File to edit/view** (Открыть файл для редактирования/просмотра) (рис. 1.7). Достаточно найти нужный файл, дважды щелкнуть по нему мышкой, и на экране появится окно для работы с этим файлом.
- **Reopen** (Переоткрыть) – вызывает подменю, содержащее список файлов, с которыми ранее работал пользователь (см. пример на рис. 1.8). Если щелкнуть по любому из файлов списка, откроется окно текстового редактора с этим файлом. Последний пункт подменю, если щелкнуть по нему мышкой, может вызвать диалоговое окно **Choose file to edit** (Выбор файла для редактирования) с еще большим списком файлов, с которыми ранее работал пользователь. В данном диалоговом окне можно удалить или открыть любой из файлов списка;

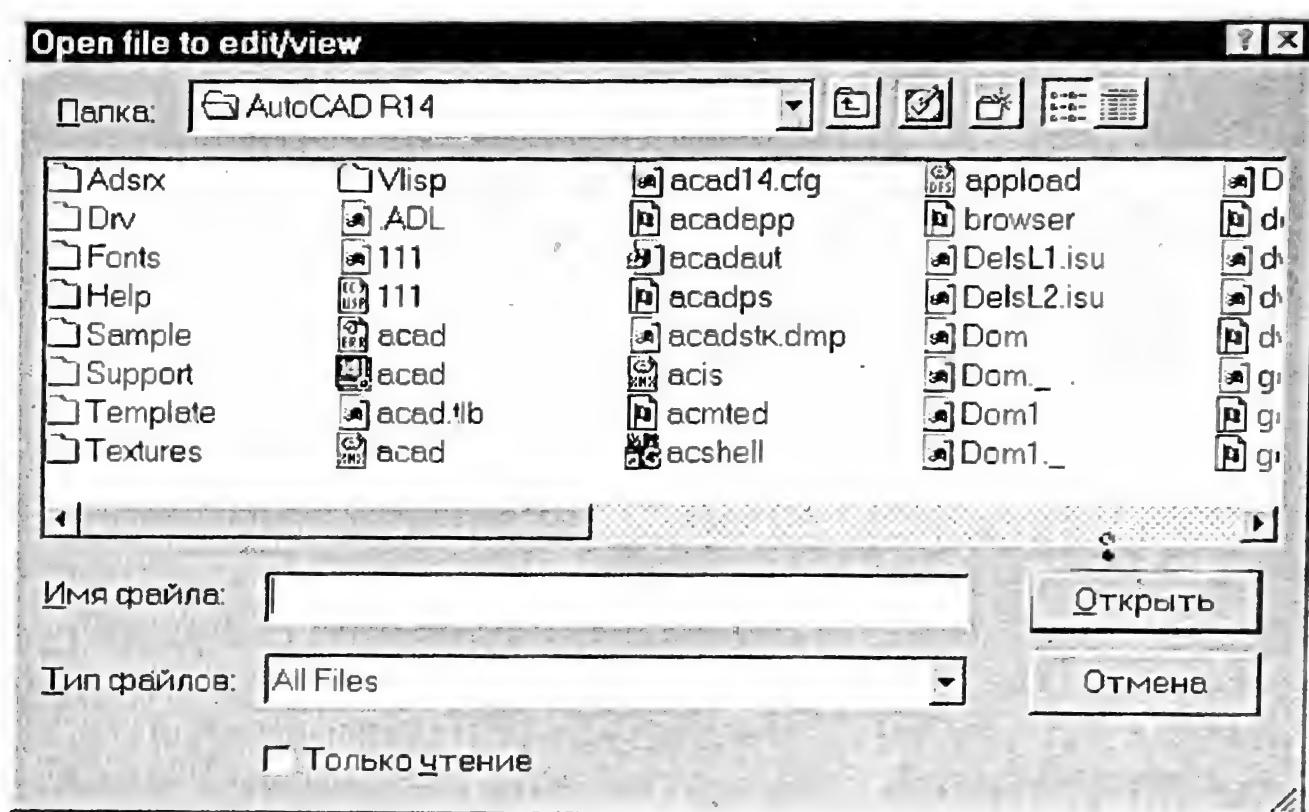


Рис. 1.7. Диалоговое окно открытия файла.

- **Save** (Сохранить) или нажатие комбинации клавиш **Ctrl+S** сохраняет файл под тем же именем;
- **Save As** (Сохранить как) или нажатие комбинации клавиш **Ctrl+Alt+S** вызывает диалоговое окно **Save as** (Сохранить как), с помощью которого можно сохранить файл под новым именем и/или в новом месте, в другом формате;
- **Save All** (Сохранить все), если он доступен, или нажатие комбинации клавиш **Alt+Shift+S** сохраняет все открытые файлы;

```

1 C:/ALISP/ADD_N_LLSP
2 C:/ALISP/COPY_LLSP
3 C:/ALISP/DIF.LSP
4 C:/ALISP/DJON.LSP
5 C:/ALISP/EARLY_A_B_LLSP
6 C:/ALISP/GCLLSP
7 C:/ALISP/KOTTEDJ.LSP
8 C:/ALISP/KOTTEDJ.MKP
9 C:/ALISP/POSL1.LSP
10 C:/ALISP/PRIVET2W.LSP
More files...

```

Рис. 1.8. Список не закрытых файлов

- **Close** (Закреть) или нажатие комбинации клавиш **Ctrl+F4** закрывает активное окно, то есть окно текущей программы (файла);
- **Revert** (Возвратить) возвращает содержимое буфера в файл;
- **Close All** (Закреть все) закрывает все окна текстового редактора;
- **Print...** (Печать) или нажатие комбинации клавиш **Ctrl+P** вызывает диалоговое окно печати для ввода имени принтера, числа копий и ряда других установок (рис. 1.9);

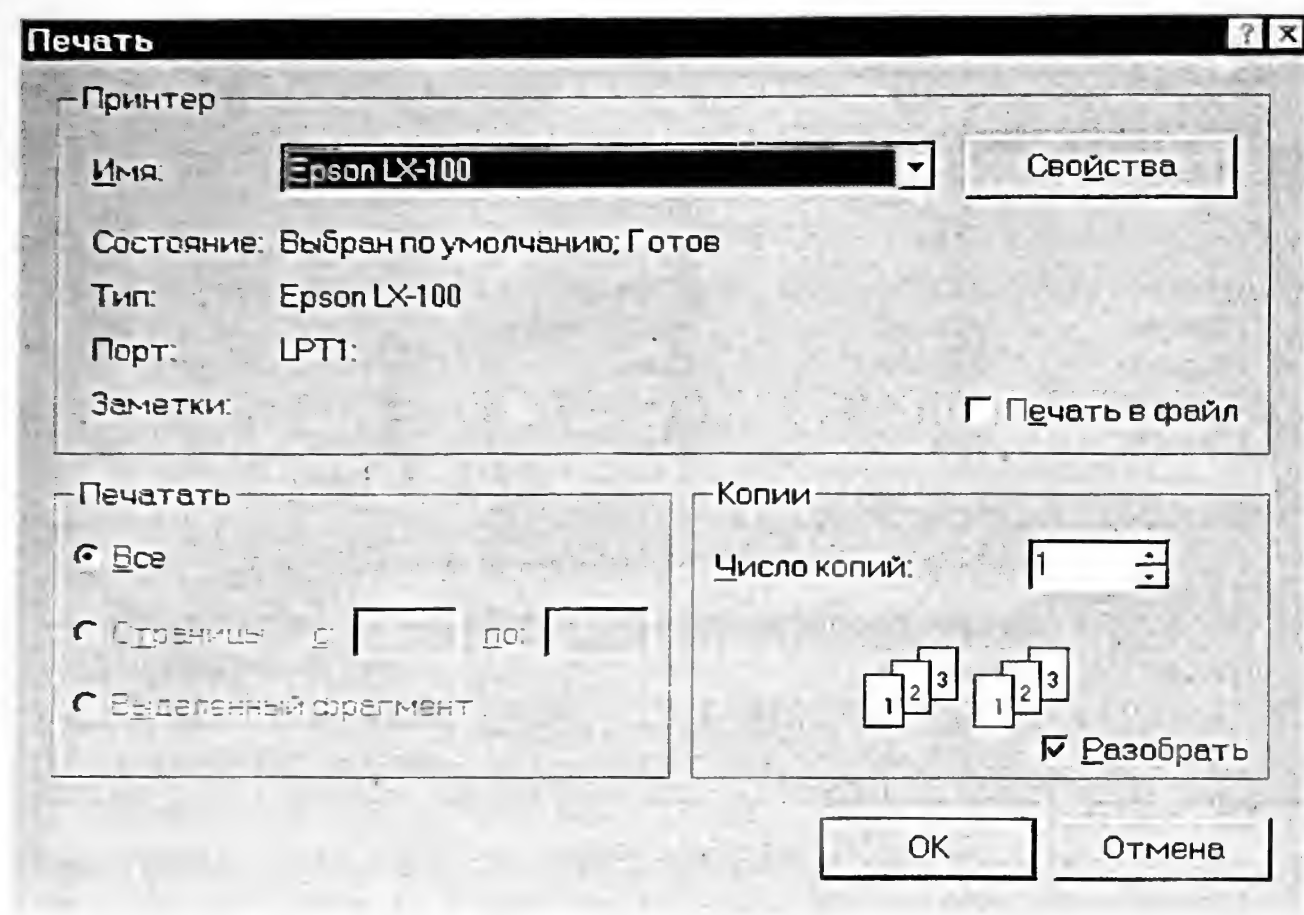


Рис. 1.9. Диалоговое окно печати

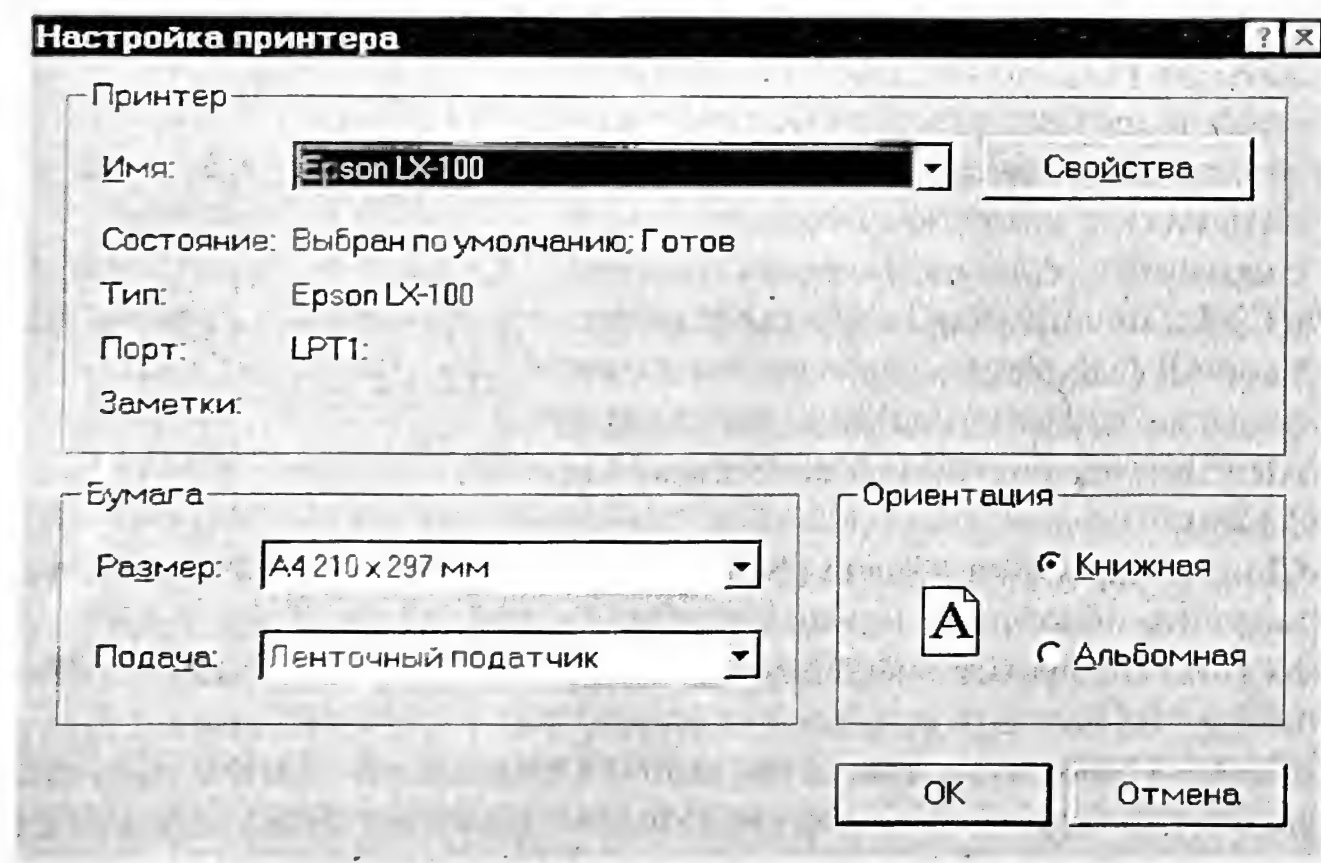


Рис. 1.10. Диалоговое окно настройки принтера

- **Print Setup...** (Настройка принтера) вызывает диалоговое окно настройки принтера для установки размера бумаги и ввода ряда других установок (рис. 1.10);
- **Make Application** (Создать приложение) или нажатие комбинации клавиш **Ctrl+Shift+M** выводит следующий список Мастеров приложений (рис. 1.11):
 - **New Application Wizard...** (Мастер нового приложения) вызывает диалоговое окно для создания нового приложения с помощью Мастера;
 - **Existing Application Wizard...** (Мастер существующего приложения) вызывает диалоговое окно для изменения существующего приложения с помощью Мастера;
 - **Build Application from Make File** (Создать приложение из make-файла) предназначено для построения приложения из файла с расширением MKP;
- **Load File...** (Загрузить файл...) или нажатие комбинации клавиш **Ctrl+Shift+L** вызывает диалоговое окно загрузки файлов **Load lisp file** (Загрузить lisp файл) (рис. 1.12);
- **Toggle Console Log...** (Установить/удалить файл регистрации консоли) вызывает диалоговое окно для создания файла регистрации **Open Log** (Открыть файл регистрации) (рис. 1.13);

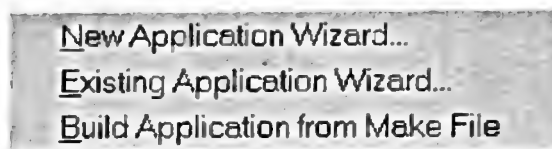


Рис. 1.11. Список Мастеров приложений

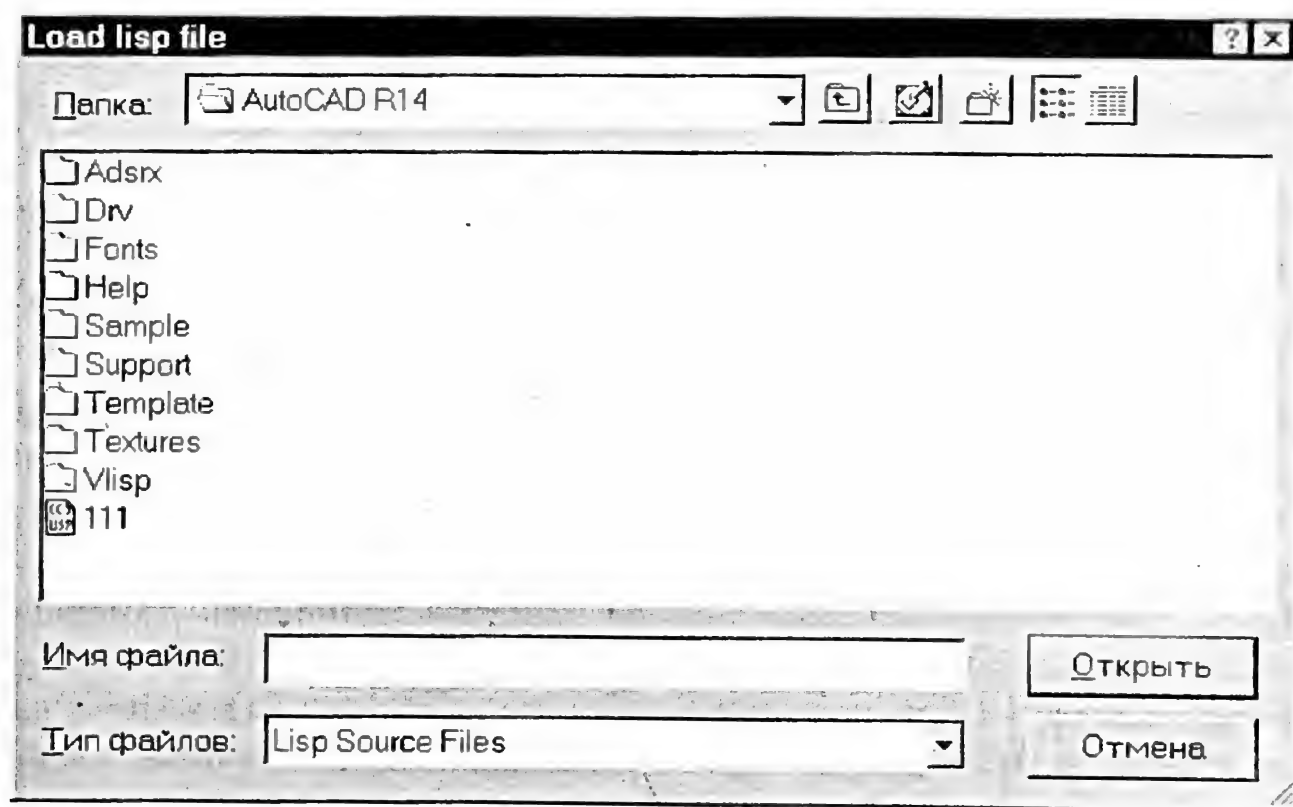


Рис. 1.12. Диалоговое окно загрузки файла

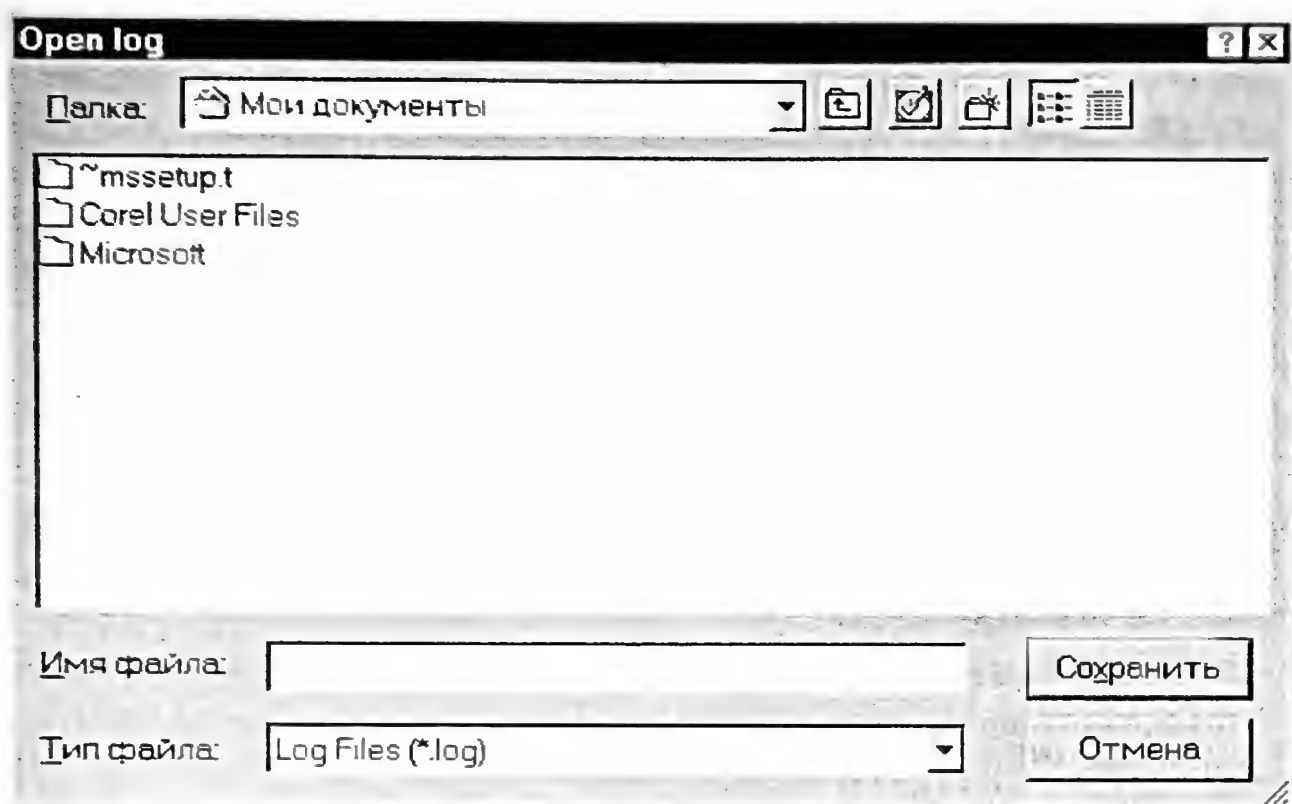


Рис. 1.13. Открытие диалогового окна файла регистрации

- **Exit** (Выйти) или нажатие комбинации клавиш **Alt+Q** вызывает диалоговое окно **Exiting Visual LISP...** (Выход из Visual LISP) для выхода из системы Visual LISP (рис. 1.14).

В диалоговом окне появляется вопрос **Really want to quit session?** (Действительно ли вы хотите выйти из Visual LISP?). Если щелкнуть мышкой по кнопке **Yes** (Да), то сеанс работы с Visual LISP завершится.

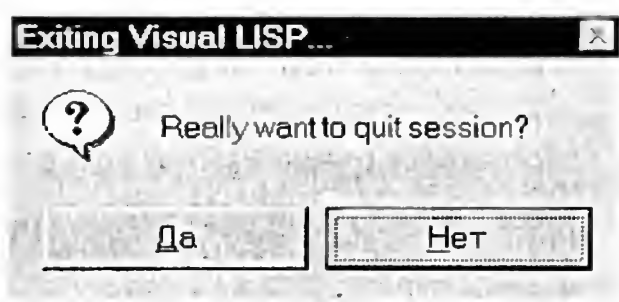


Рис. 1.14. Диалоговое окно выхода из системы Visual LISP

1.2.2. Падающее меню редактирования: **Edit**

Выбор пункта **Edit** (Редактирование) главного меню или нажатие комбинации клавиш **Alt+E** вызывает падающее меню редактирования (рис. 1.15).

Падающее меню редактирования включает большой набор пунктов:

- **Undo** (Отменить) или нажатие комбинации клавиш **Ctrl+Z** отменяет последнюю из выполненных команд;
- **Redo** (Вернуть) или нажатие комбинации клавиш **Ctrl+Alt+Z** дает возможность повторно выполнить последнюю из отмененных команд;
- **Cut** (Вырезать) или нажатие комбинации клавиш **Ctrl+X** удаляет выделенный фрагмент текста и помещает его в буфер обмена;

Undo	Ctrl-Z
Redo	Ctrl-Alt-Z
Cut	Ctrl-X
Copy	Ctrl-C
Paste	Ctrl-V
Delete	Del
Select All	Ctrl-A
Parentheses Matching	Ctrl-M ▶
Extra Commands	Ctrl-E ▶

Undo	Ctrl-Z
Redo	Ctrl-Alt-Z
Cut	Ctrl-X
Copy	Ctrl-C
Paste	Ctrl-V
Delete	Del
Select All	Ctrl-A
Clear Console Window	
Clear Console Input	Esc
Keep but Ignore Input	Shift-Esc
Parentheses Matching	Ctrl-M ▶
Console History Up	Tab
Console History Down	Shift-Tab

Рис. 1.15. Падающее меню редактирования Edit при активном окне текстового редактора (слева) и при активном окне консоли Visual LISP (справа)

- **Copy** (Копировать) или нажатие комбинации клавиш **Ctrl+C** копирует выделенный фрагмент текста и помещает его в буфер обмена;
- **Paste** (Вставить) или нажатие комбинации клавиш **Ctrl+V** вставляет содержимое буфера обмена в текущую позицию курсора в окне текстового редактора или окне консоли;
- **Delete** (Удалить) или нажатие клавиши **Del** удаляет выделенный фрагмент программы;
- **Clear Console Window** (Очистить окно консоли) очищает окно консоли, удаляет всю имеющуюся в окне информацию;
- **Clear Console Input** (Очистить ввод с консоли) или нажатие клавиши **Esc** очищает введенную пользователем информацию в окне консоли;
- **Keep but Ignore Input** (Сохранить, но игнорировать ввод) осуществляет сохранение ранее введенной информации и переход в состояние ожидания ввода данных;
- **Parentheses Matching** (Соответствие круглых скобок) или нажатие комбинации клавиш **Ctrl+M** определяет соответствие круглых скобок. При выборе этой команды появляется следующий список возможных действий по поиску круглых скобок (рис. 1.16):
- **Match Forward** (Поиск закрывающей скобки) или нажатие комбинации клавиш **Ctrl+]** осуществляет поиск соответствующей закрывающей круглой скобки;

Match Forward	Ctrl-]
Match Backward	Ctrl-[
Select Forward	Ctrl-Shift-]
Select Backward	Ctrl-Shift-[

Рис. 1.16. Список команд для поиска круглых скобок

- **Match Backward** (Поиск открывающей скобки) или нажатие комбинации клавиш **Ctrl+[** осуществляет поиск соответствующей открывающей круглой скобки;
- **Select Forward** (Выделение выражения после скобки) или нажатие комбинации клавиш **Ctrl+Shift+]** производит выделение выражения, которое начинается с указанной открывающей скобки;
- **Select Backward** (Выделение выражения до скобки) или нажатие комбинации клавиш **Ctrl+Shift+[** производит выделение выражения, которое заканчивается указанной закрывающей скобкой;
- **Console History UP** (Хронология сообщений консоли вверх) или нажатие клавиши **Tab** вызывает отображение хронологии сообщений до текущего сообщения;
- **Console History Down** (Хронология сообщений консоли вниз) или нажатие комбинации клавиш **Shift+Tab** вызывает отображение хронологии сообщений после текущего сообщения;
- **Extra Commands** (Особые команды) или нажатие комбинации клавиш **Ctrl+E** позволяет вызвать список особых команд.

1.2.3. Падающее меню поиска: Search

Выбор пункта **Search** (Поиск) главного меню или нажатие комбинации клавиш **Alt+S** вызывает падающее меню поиска (рис. 1.17).

Меню, представленное на рис. 1.17, содержит следующие пункты:

- **Find...** (Найти...) или нажатие комбинации клавиш **Ctrl+F** выводит диалоговое окно с текстовым полем **Find What:** (Найти что);
- **Replace...** (Заменить...) или нажатие комбинации клавиш **Ctrl+H** выводит диалоговое окно **Find** (Найти), в котором имеется два текстовых поля **Find What** (Найти что) и **Replace with:** (Заменить на);
- **Find/Replace Next** (Найти/Заменить следующее) или нажатие клавиши **F3** находит в программе старое сочетание символов и заменяет его на новое;
- **Complete Word by Match** (Завершить слово в полном соответствии) или нажатие комбинации клавиш **Ctrl+Space** дописывает имена встроенных функций, системных переменных;

Find...	Ctrl-F
Replace...	Ctrl-H
Find/Replace Next	F3
Complete Word by Match	Ctrl-Space
Complete Word by ApropoS	Ctrl-Shift-Space
Bookmarks	
First message	Shift-F11
Next message	F11
Go to Line...	Ctrl-G
Go to Last Edited	Ctrl-Shift-G

Рис. 1.17. Падающее меню поиска Search

- **Complete Word by Apropos** (Поиск слова по фрагменту) или нажатие комбинации клавиш **Ctrl+Shift+Space** вызывает список имен функций, системных переменных, содержащих введенный фрагмент. При щелчке по нужному имени это имя заменяет введенный фрагмент;
- **Bookmarks** (Установить закладки) выводит всплывающее меню для установки закладок;
- **Go to Line...** (Идти к строке...) или нажатие комбинации клавиш **Ctrl+G** выводит диалоговое окно **Enter line number (fixnum)** (Введите номер строки) для установки номера строки;
- **Go to Last Edited** (Идти к последнему месту редактирования) или нажатие комбинации клавиш **Ctrl+Shift+G** устанавливает курсор в месте последнего редактирования.

1.2.4. Падающее меню просмотра: View

Выбор пункта **View** (Просмотр) главного меню или нажатие комбинации клавиш **Alt+V** вызывает падающее меню просмотра (рис. 1.18).

Меню, представленное на рис. 1.18, содержит следующие пункты:

- **Inspect** (Проверить) или нажатие комбинации клавиш **Ctrl+Shift+I** вызывает диалоговое окно **Enter expression to inspect** (Введите выражение для проверки) для контроля за правильностью работы проверяемого выражения;
- **Trace Stack** (Стек трассировки) или нажатие комбинации клавиш **Ctrl+Shift+T** вызывает окно **Trace Stack** с результатами трассировки стека;
- **Error Trace** (Трассировка ошибки) или нажатие комбинации клавиш **Ctrl+Shift+R** вызывает окно **Error trace** с результатами трассировки ошибки;
- **Symbol Service** (Обслуживание символов) или нажатие комбинации клавиш **Ctrl+Shift+S** вызывает диалоговое окно **Enter symbol name** (Введите имя символа) для обслуживания символов;
- **Watch Window** (Окно наблюдений) или нажатие комбинации клавиш **Ctrl+Shift+W** вызывает окно **Watch** (Наблюдения) для просмотра текущих значений символов;
- **Apropos Window** (Окно поиска по фрагменту) или нажатие

Inspect...	Ctrl-Shift-I
Trace Stack	Ctrl-Shift-T
Error Trace	Ctrl-Shift-R
Symbol Service...	Ctrl-Shift-S
Watch Window	Ctrl-Shift-W
Apropos Window...	Ctrl-Shift-A
Breakpoints Window	Ctrl-Shift-B
Output Window	Ctrl-Shift-O
LISP Console	F6
Browse Drawing Database	
Toolbars...	

Рис. 1.18. Падающее меню просмотра View

комбинации клавиш **Ctrl+Shift+A** открывает диалоговое окно **Apropos...** (Поиск по фрагменту) для настройки системы поиска слова по фрагменту;

- **LISP Console** (Консоль LISP) или нажатие клавиши **F6** вызывает окно **Visual LISP Console** (Окно консоли Visual LISP);
- **Browse Drawing Database** (Обзор базы данных рисунков) вызывает команды просмотра базы данных рисунков, всех объектов, таблиц и т.д.;
- **Toolbars...** (Панели инструментов) вызывает диалоговое окно установки необходимых панелей инструментов.

1.2.5. Падающее меню управления проектом: *Project*

Выбор пункта **Project** главного меню или нажатие комбинации клавиш **Alt+P** вызывают падающее меню управления проектом (рис. 1.19).

Меню, представленное на рис. 1.19, включает в себя следующие пункты:

- **New Project** (Новый проект) обеспечивает создание нового проекта;
- **Open Project** (Открыть проект) или нажатие комбинации клавиш **Ctrl+Shift+P** обеспечивает открытие существующего проекта.

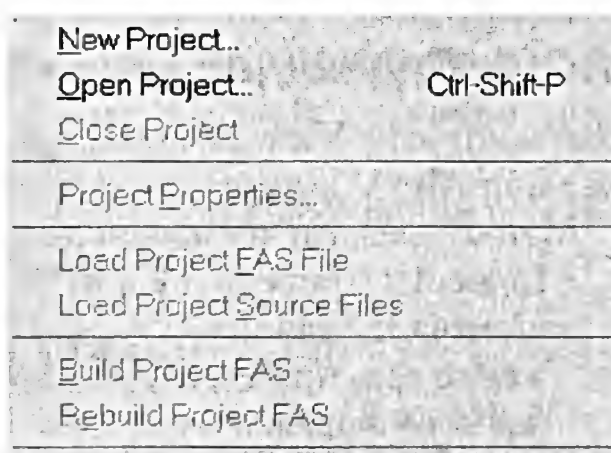


Рис. 1.19. Падающее меню управления проектом *Project*

1.2.6. Падающее меню отладки: *Debug*

Выбор пункта **Debug** главного меню или нажатие комбинации клавиш **Alt+D** вызывает падающее меню отладки (рис. 1.20).

Меню, представленное на рис. 1.20, содержит следующие пункты:

- **Add Watch** (Добавить наблюдение) вызывает диалоговое окно **Enter expression to watch** (Введите выражение для наблюдений) значений введенного выражения;
- **Watch Last Evaluation** (Наблюдение последних вычислений) вызывает окно **Watch** (Наблюдения) для просмотра последних вычислений;
- **Trace Command** (Трассировка команд) включает или выключает трассировку команд;
- **Stop Once** (Остановка сразу) включает или выключает немедленную остановку вычислений;
- **Break On Error** (Прерывание на ошибке) включает или выключает прерывание выполнения при обнаружении ошибки;
- **Animate** (Оживить) включает или выключает выполнение вычислений.

1.2.7. Падающее меню средств управления системой: Tools

Выбор пункта Tools (Инструменты) главного меню или нажатие комбинации клавиш **Alt+T** вызывает падающее меню средств управления системой (рис. 1.21).

В меню представлены следующие пункты:

- **Load Selection** (Загрузить выделенное) или нажатие комбинации клавиш **Ctrl+Shift+E** позволяет загрузить выделенное выражение на выполнение и выполнить его;
- **Load Text in Editor** (Загрузить текст из окна редактора) или нажатие комбинации клавиш **Ctrl+Alt+E** позволяет загрузить текст из окна текстового редактора на исполнение;
- **Check Selection** (Проверить выделенное) или нажатие комбинации клавиш **Ctrl+Shift+C** позволяет проверить и загрузить выделенное выражение на выполнение и выполнить его;
- **Check Text in Editor** (Проверить текст в окне редактора) или нажатие

Step Into	F8
Step Over	Shift-F8
Step Out	Ctrl-Shift-F8
Continue	Ctrl-F8
Reset to Top Level	Ctrl-P
Quit Current Level	Ctrl-Q
Add Watch...	Ctrl-W
Watch Last Evaluation	
Toggle Breakpoint	F9
Clear All Breakpoints	Ctrl-Shift-F9
Last Break Source	Ctrl-F9
Trace Command	
Stop Once	
Break On Error	
Animate	
Abort Evaluation	

Рис. 1.20. Падающее меню отладки Debug

Load Selection	Ctrl-Shift-E
Load Text in Editor	Ctrl-Alt-E
Check Selection	Ctrl-Shift-C
Check Text in Editor	Ctrl-Alt-C
Format AutoLISP in Selection	Ctrl-Shift-F
Format AutoLISP in Editor	Ctrl-Alt-F
Interface Tools	▶
Window Attributes	▶
Environment Options	▶
Save Settings	

AutoCAD Mode	Ctrl-Shift-C
Window Attributes	▶
Environment Options	▶
Save Settings	

Рис. 1.21. Падающее меню Tools при активном окне текстового редактора (слева) и при активном окне консоли Visual LISP (справа)

комбинации клавиш **Ctrl+Alt+C** позволяет проверить и загрузить текст из окна текстового редактора на исполнение;

- **AutoCAD Mode** (Режим AutoCAD) или нажатие комбинации клавиш **Ctrl+Shift+C** позволяет переключиться в режим работы AutoCAD;
- **Window Attributes** (Параметры окна) позволяет изменить следующие параметры окна Visual LISP (рис. 1.22):
 - **Configure Current...** (Текущая конфигурация) обеспечивает вызов диалогового окна установки текущей конфигурации окна Visual LISP;
 - **Font...** (Шрифты...) обеспечивает вызов диалогового окна выбора шрифта окна Visual LISP;
- **Environment Options** (Режимы среды) позволяет установить режимы работы среды Visual LISP (рис. 1.23);
- **Save Setting** (Сохранить установки) обеспечивает сохранение установок рабочей среды Visual LISP.

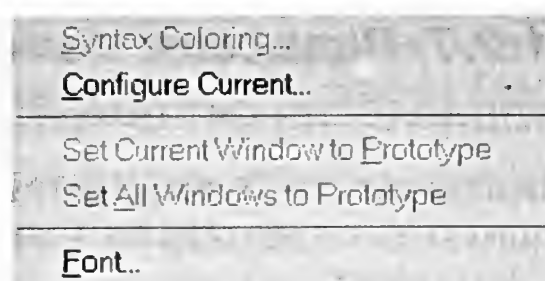


Рис. 1.22. Список параметров окна Visual LISP

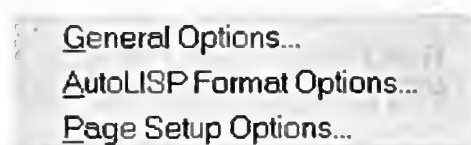


Рис. 1.23. Список режимов среды

1.2.8. Падающее меню управления окнами: Window

Visual LISP позволяет организовать работу с несколькими окнами одновременно. Выбор пункта **Window** (Окно) главного меню или нажатие комбинации клавиш **Alt+W** вызывают падающее меню управления работой с несколькими окнами (рис. 1.24).

Меню, представленное на рис. 1.24, включает в себя следующие пункты:

- **Tile Horizontally** (Горизонтальные подокна) обеспечивает горизонтальное размещение окон, которые располагаются одно под другим сверху вниз. Всем окнам отводятся равные части рабочего стола;
- **Tile Vertically** (Вертикальные подокна) обеспечивает вертикальное расположение окон. Окна располагаются слева направо и им отводятся равные части рабочего стола;
- **Cascade** (Каскад) обеспечивает каскадное

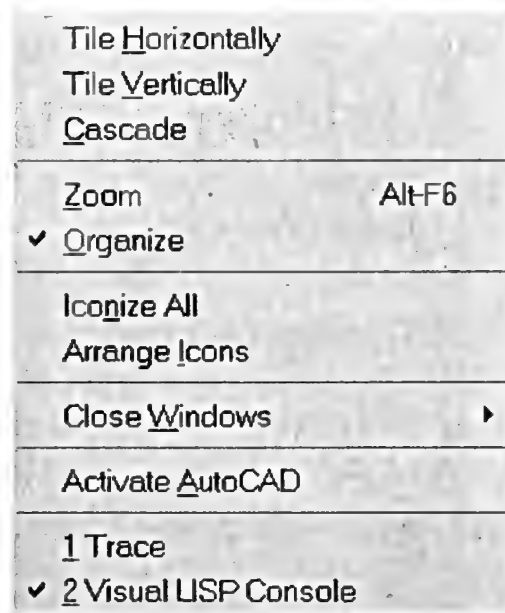


Рис. 1.24. Падающее меню управления окнами Window

расположение окон. Окона располагаются друг за другом уступом так, что заголовки всех окон остаются видны. Активное окно размещается поверх остальных;

- **Zoom** (Масштабировать) обеспечивает уменьшение/увеличение размера отображения активного окна;
- **Organize** (Организовать) определяет режим упорядочения расположения окон, работает как переключатель;
- **Iconize All** (Минимизировать все) минимизирует все открытые окна Visual LISP;
- **Arrange Icons** (Упорядочить минимизированные окна) упорядочивает все минимизированные окна Visual LISP – иконки;
- **Close Windows** (Заккрыть окна) обеспечивает закрытие окон: **All** (Все) или **Inspectors** (Проверки);
- **Activate AutoCAD** (Активизировать AutoCAD) активизирует систему AutoCAD.

На рабочем столе Visual LISP могут быть представлены несколько открытых окон, например, окно трассы **Trace**, окно консоли **Visual LISP Console** и несколько окон текстового редактора.

1.2.9. Падающее меню помощи: Help

Выбор пункта **Help** главного меню или нажатие комбинации клавиш **Alt+N** вызывает падающее меню помощи (рис. 1.25).

Падающее меню помощи содержит следующие пункты:

- **Visual LISP Help Topics** (Справочная система) или нажатие клавиши **F1** обеспечивает вызов справочной системы по Visual LISP (рис. 1.26);
- **Tutorial** (Обучающая программа) вызывает краткий учебник по Visual LISP;
- **About Visual LISP** (О программе Visual LISP) обеспечивает дополнительной информацией по Visual LISP.

Диалоговое окно справочной системы по Visual LISP имеет три закладки: **Содержание**, **Указатель** и **Поиск**.

В окне **Содержание** представлен список названий разделов справочной системы Visual LISP. Щелкая мышкой по названиям того или иного раздела, можно получить исчерпывающую информацию по Visual LISP.

В окне **Указатель** содержится список функций и терминов. Для получения нужной информации введите первые буквы нужной функции в первом поле справочной системы или выберите ее с помощью полосы прокрутки.

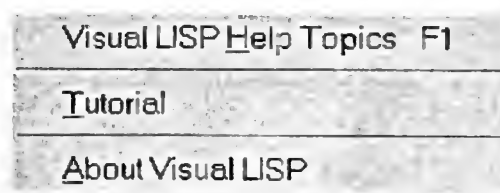


Рис. 1.25. Падающее меню помощи Help

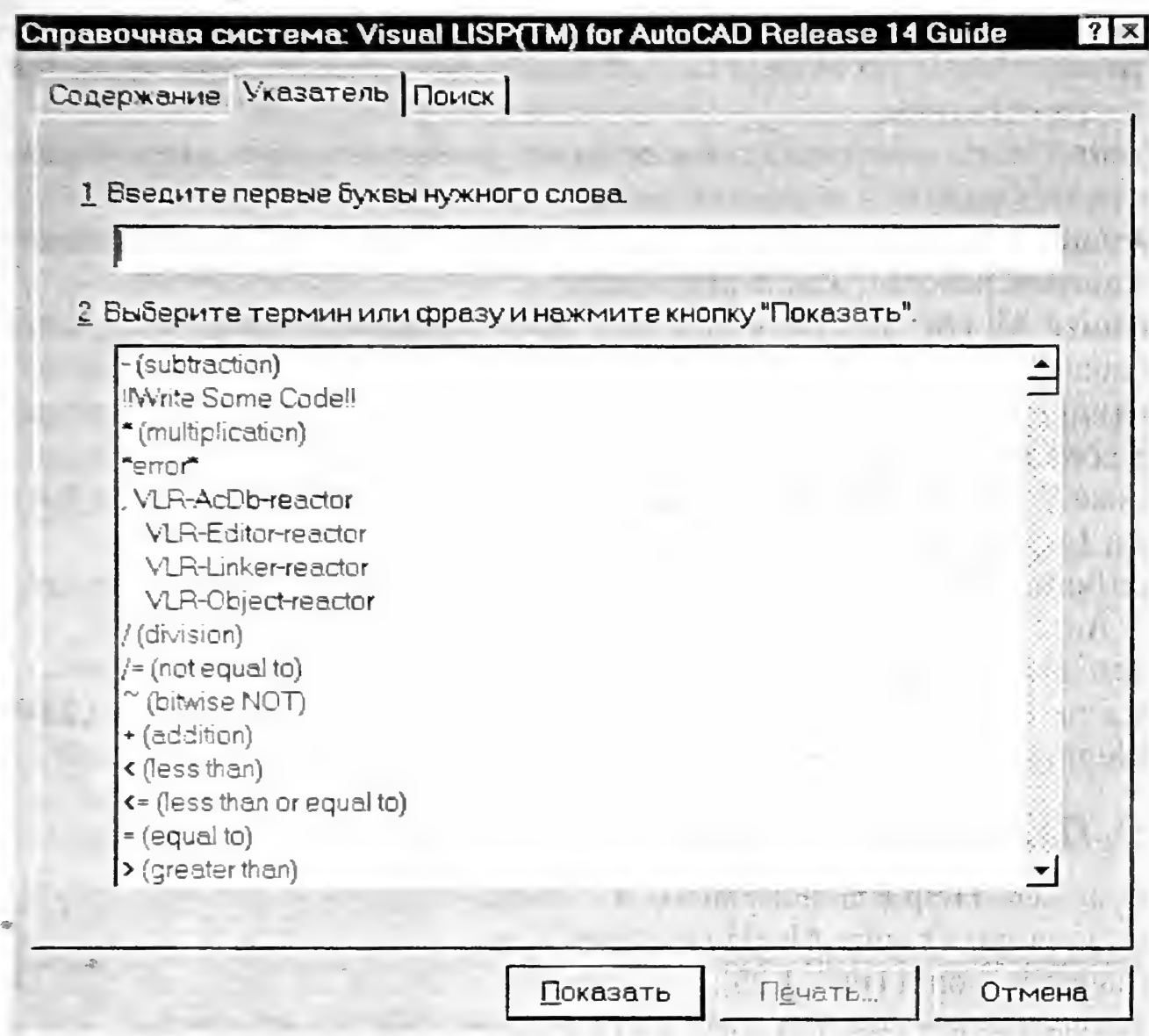


Рис. 1.26. Справочная система для Visual LISP

Окно **Поиск** предназначено для оперативного поиска нужной информации. Чтобы получить необходимые сведения, достаточно ввести в верхнем поле искомый термин, в нижнем поле выбрать вариант использования этого термина (для сужения диапазона поиска) и щелкнуть по кнопке **Показать**.

1.3. Панели инструментов Visual LISP

Для облегчения работы со средой Visual LISP можно вывести на экран инструментальные панели (рис. 1.27). Вывести или отключить требуемые панели инструментов можно с помощью пункта **Toolbars...** (Панели инструментов) падающего меню **View** (Просмотр), который вызывает диалоговое окно выбора панелей инструментов **Toolbars** (рис. 1.28).

Диалоговое окно обеспечивает управление отображением на экране следующих панелей инструментов: **Standard** (Стандартная), **Search** (Поиск),

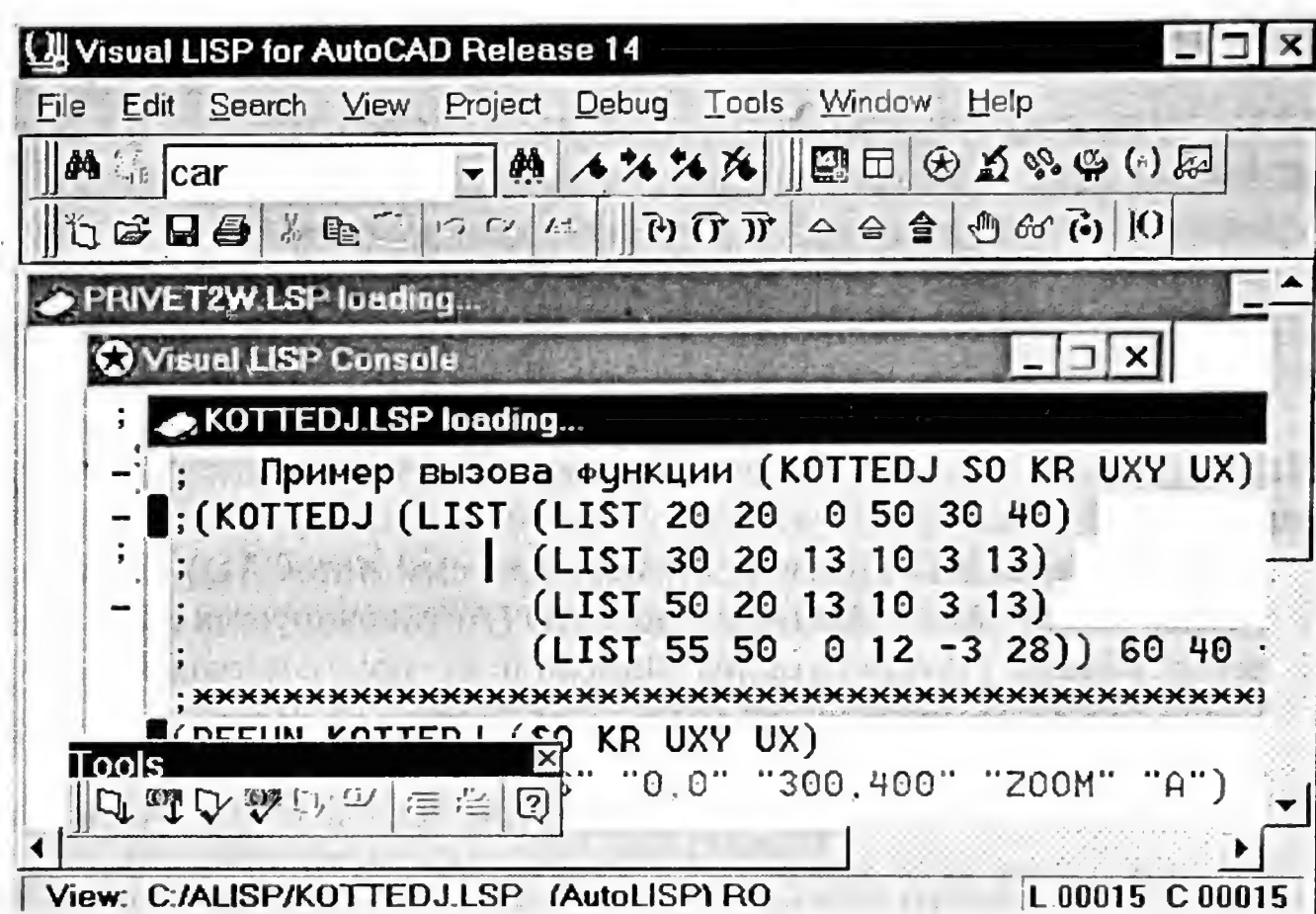


Рис. 1.27. Главное окно Visual LISP с инструментальными панелями

Tools (Инструменты), **Debug** (Отладка), **View** (Просмотр). Для того чтобы вывести ту или иную панель на экран, необходимо щелчком мышки включить/выключить соответствующие флажки, после чего нажать кнопку **Apply** (Применить). Если щелкнуть по кнопке **Show all** (Показать все), в главном окне Visual LISP появятся все инструментальные панели.

Для удаления инструментальных панелей из главного окна Visual LISP необходимо нажать кнопку **Hide all** (Скрыть все), а затем кнопку **Close** (Заккрыть).

При перемещении указателя мышки по значкам (пиктограммам) инструментальных панелей высвечивается назначение каждой кнопки панели, а в строке состояния в самом низу главного окна Visual LISP появляется более подробное объяснение. С каждой кнопкой связан определенный пункт меню, а рисунок на этой кнопке передает суть этого пункта меню. Вызывать пункт меню с помощью кнопки гораздо быстрее, чем выбирать в меню.

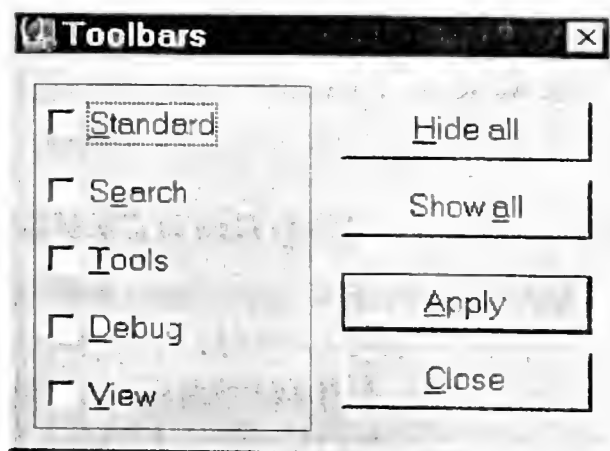


Рис. 1.28. Диалоговое окно выбора панелей инструментов









1.3.1. Панель инструментов просмотра: View

Панель инструментов просмотра включает восемь основных кнопок (рис. 1.29).



Рис. 1.29. Панель инструментов просмотра View

На панели инструментов **View** (Просмотр) размещаются следующие кнопки:

-  **Activate AutoCAD** (Активизировать систему AutoCAD). Описание в строке состояний – Activate AutoCAD (Активизируется AutoCAD);
-  **Select window** (Выбрать окно). Описание в строке состояний – **Select Visual LISP window from list** (Выбор окна Visual LISP из списка);
-  **LISP console** (Выбрать консоль AutoLISP). Описание в строке состояний – Activate Visual LISP console / bring console to top (Активизирование Visual LISP консоли / перенесение консоли наверх);
-  **Inspect** (Проверить). Описание в строке состояний – Open an inspector window and enter expression (Открытие окна просмотра для ввода выражения);
-  **Trace** (Трассировка ошибки). Описание в строке состояний – Open the inspector window for last error trace stack (Открытие окна просмотра последней ошибки трассы стека);
-  **Symbol service** (Обслужить символы). Описание в строке состояний – Activate the symbol service window (Активизация окна обслуживания символов);
-  **Apropos** (Найти слово по фрагменту). Описание в строке состояний – Open the apropos window for assistance on LISP expressions (Открытие окна поиска слова по фрагменту для помощи LISP-выражениям);
-  **Watch window** (Открыть окно наблюдений). Описание в строке состояний – Open the watch window (Открытие окна наблюдений).

1.3.2. Панель инструментов: Tools

Панель инструментов включает девять кнопок (рис. 1.30).

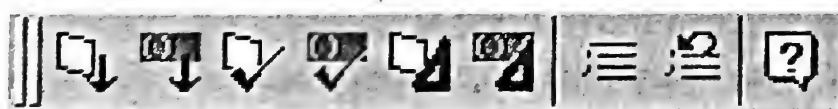











Рис. 1.30. Панель инструментов Tools

На панели инструментов Tools (Инструменты) размещаются следующие кнопки:

-  **Load activate edit window** (Загрузить программу из активного окна редактирования). Описание в строке состояний – Load and evaluate the code in the active editor window (Загрузка и выполнение программы в активном окне редактора);
-  **Load selection** (Загрузить выделенное выражение). Описание в строке состояний – Load and evaluate the selected text (Загрузка и выполнение выделенного текста);
-  **Check edit window** (Проверить синтаксис программы в активном окне редактирования). Описание в строке состояний – Perform syntax checking of AutoLISP code in the active editor window (Выполнение синтаксической проверки программы AutoLISP в активном окне редактора);
-  **Check selection** (Проверить синтаксис выделенного). Описание в строке состояний – Perform syntax checking of selected AutoLISP code in edit window (Выполнение проверки синтаксиса выделенной программы AutoLISP в окне редактирования);
-  **Format edit window** (Отформатировать программу в активном окне редактирования). Описание в строке состояний – Format AutoLISP code in the active editor window (Форматирование программы AutoLISP в активном окне редактора);
-  **Format selection** (Отформатировать выделенный фрагмент). Описание в строке состояний – Format AutoLISP code in the current selection (Форматирование программы AutoLISP в текущем выборе);
-  **Comment block** (Закомментировать блок). Описание в строке состояний – Comment out the selection block of text (Комментирование выбранного блока текста);
-  **Uncomment block** (Разкомментировать блок). Описание в строке состояний – Uncomment the selection block of text (Отмена комментариев у выбранного блока текста);
-  **Help** (Помощь). Описание в строке состояний – Visual LISP Help (Помощь Visual LISP).

1.3.3. Панель инструментов отладки: Debug

Панель инструментов отладки включает девять кнопок и один индикатор в виде кнопки (последний на панели инструментов) (рис. 1.31).

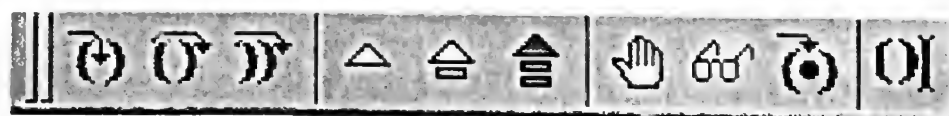











Рис. 1.31. Панель инструментов отладки Debug

На индикаторе изображены круглые скобки и положение курсора. На индикаторе курсор может стоять перед круглыми скобками; это означает, что процесс отладки остановлен перед выражением **Stopped before** (Остановлен перед) или после – **Stopped after** (Остановлен после).

На панели инструментов **Debug** (Отладка) размещаются следующие кнопки:

-  **Step into** (Шагнуть внутрь выражения). Описание в строке состояний – Step into the nested expression (Шагнуть внутрь вложенного выражения);
-  **Step over** (Шагнуть с обходом). Описание в строке состояний – Step over the next expression (Шагнуть до следующего выражения);
-  **Step out** (Шагнуть вне выражения). Описание в строке состояний – Step out to the end of the current function (Шагнуть вне к концу текущей функции);
-  **Continue** (Продолжить отладку). Описание в строке состояний – Continue to the next breakpoint or to the end (Продолжить до следующей точки прерывания или до конца);
-  **Quit** (Выйти из текущего уровня отладки). Описание в строке состояний (внизу окна) – Quit the current debug level (Выход из текущего уровня отладки);
-  **Reset** (Вернуться на верхний уровень). Описание в строке состояний – Reset to the top level (Вернуться на верхний уровень);
-  **Toggle breakpoint** (Установить/удалить точку прерывания). Описание в строке состояний – Add or remove a breakpoint at the current location (Добавление или удаление точек прерываний в текущем положении);
-  **Add watch** (Добавить выражение для просмотра). Описание в строке состояний – Add an expression to watch (Добавить выражение для просмотра);
-  **Last break** (Последнее прерывание). Описание в строке состояний – Show last break source (Показать источник последнего прерывания).

1.3.4. Панель инструментов поиска: **Search**








Панель инструментов поиска включает семь кнопок и раскрывающийся список с текстовым полем (рис. 1.32).



Рис. 1.32. Панель инструментов поиска **Search**

В текстовом поле раскрывающегося списка вводятся символы для поиска их в активном окне текстового редактора или окне консоли. Щелчком по стрелке, направленной вниз, можно раскрыть список и выбрать в нем ранее введенные символы.

На панели инструментов **Search** (Поиск) размещаются следующие кнопки:

-  **Find** (Найти). Описание в строке состояний – Search for text (Поиск текста). Точнее, поиск введенного текста в текстовом поле диалогового окна Find, которое появляется после щелчка по кнопке **Find** (Найти);
-  **Replace** (Заменить). Описание в строке состояний – Search for text & replace it with new text (Поиск введенного текста и замена его на новый текст);
-  **Find toolbars string** (Найти строку текстового поля). Описание в строке состояний – Find the current string in the toolbars (Найти текущую строку текстового поля);
-  **Toggle bookmark** (Установить/удалить закладку). Описание в строке состояний – Add/remove a bookmark at the current position (Добавление/удаление закладки на текущей позиции);
-  **Next bookmark** (Перейти к следующей закладке). Описание в строке состояний – Move to the next bookmark (Перейти к следующей закладке);
-  **Previous bookmark** (Перейти к предыдущей закладке). Описание в строке состояний – Move to the previous bookmark (Перейти к предыдущей закладке);
-  **Clear all bookmark** (Удалить все закладки). Описание в строке состояний – Clear all bookmark (Удалить все закладки).

Информация, отображаемая в строке состояния (в самом низу главного окна Visual LISP), изменяется в зависимости от того, какое действие в настоящий момент вы выполняете в Visual LISP.

Обратите внимание на минимизированное окно поиска нескольких символов. При запуске это окно информирует о текущем состоянии Visual LISP. Кроме того, оно может содержать дополнительную информацию, например, об ошибках VLISP во время запуска.

1.4. Текстовый редактор Visual LISP

Текстовый редактор Visual LISP предназначен для эффективной разработки, проверки и отладки программ на AutoLISP.

Окно текстового редактора вызывается автоматически при открытии файла с программой на AutoLISP в диалоговом окне **Open file to edit/view** (Открыть файл для редактирования/просмотра) (см. рис. 1.7).

Вызвать диалоговое окно открытия файла можно двумя способами:

- выбрать пункт **Open File...** (Открыть файл) падающего меню **File** (Файл) (см. рис. 1.5);
- нажать комбинацию клавиш **Ctrl+O**.

В диалоговом окне открытия файла необходимо найти соответствующую папку (каталог) и дважды щелкнуть по ней мышкой. В раскрытой папке (каталоге) аналогичным способом следует открыть нужный файл, который будет помещен в текстовое окно системы Visual LISP.

В качестве примера рассмотрим файл DOM.LSP, который содержит программу, имеющую всего одну функцию языка AutoLISP – (**COMMAND...**) (рис. 1.33).

Когда вы вносите изменения в тексте программы в редакторе или добавляете новый текст, рядом с именем файла в строке состояния появляется звездочка (*). Она сохраняется рядом с названием файла до тех пор, пока вы не сохраните внесенные изменения либо не закроете файл.

Следует помнить, что одновременно можно работать только с одним файлом в одном окне текстового редактора. Каждый раз при открытии файла Visual LISP отображает файл в новом окне текстового редактора.

Текстовый редактор Visual LISP обеспечивает цветное отображение программ. Кроме того, он выполняет синтаксический анализ программ на языке AutoLISP и назначает соответствующие цвета различным константам, именам функций. Это позволяет выявлять синтаксические ошибки до выполнения компиляции.

Текстовый редактор предоставляет возможность форматировать программы AutoLISP, тем самым делая их более удобными для чтения. При этом можно выбирать различные стили форматирования.

Программы на языке AutoLISP содержат большое количество круглых скобок. Текстовый редактор помогает найти ближайшую закрывающую круглую скобку, которая соответствует указанной открывающей скобке.

Путем высвечивания текстовый редактор Visual LISP позволяет выбирать отдельные фрагменты программы и выполнять их. При этом необязательно выходить из текстового редактора и выполнять всю программу в целом. Помимо этого, у пользователя есть возможность осуществлять поиск слова или выражения, проверять синтаксис отдельных выделенных фрагментов программы.

Текстовый редактор позволяет загрузить программу для последующего ее выполнения. В таком случае важно, чтобы окно текстового редактора было активным. Если вы не уверены в этом, щелкните мышкой в любом месте окна и сделайте его активным. Чтобы загрузить программу, щелкните мышкой по кнопке **Load activate edit window** (Загрузить программу из

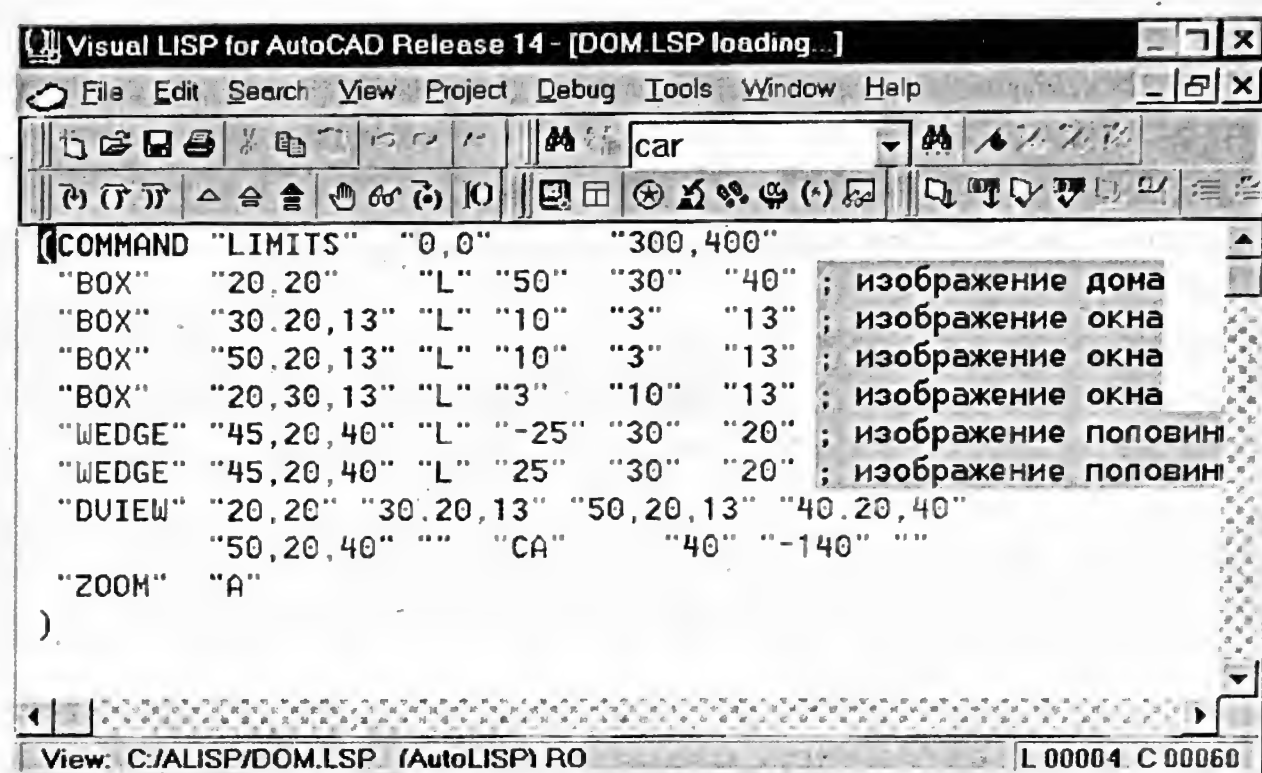


Рис. 1.33. Окно текстового редактора

активного окна редактирования) или выберите пункт **Load Text in Editor** (Загрузить текст программы из активного окна редактирования) падающего меню **Tools** (Инструменты) главного меню Visual LISP. На экране появится сообщение о загрузке программы в окне консоли Visual LISP, такое как, например, на рис. 1.34.

Из сообщения, приведенного на рис. 1.34, следует, что загрузка выполнена из текстового редактора файла DOM.LSP, который состоит из одной формы с указанием имени файла и каталогов (пути), в которых он находится.

После загрузки программы ее можно запустить на выполнение из командной строки консоли. Для этого достаточно ввести имя программы в круглых скобках и нажать клавишу **Enter**.

Во время ввода команд с консоли Visual LISP или выполнения программы, загруженной из текстового редактора, можно переключаться между окнами AutoCAD и Visual LISP. Помимо стандартных методов переключения между приложениями Windows, существует еще несколько методов переключения между окнами AutoCAD и Visual LISP.

Если вы находитесь в Visual LISP, используйте пункт **Activate AutoCAD** (Активизировать AutoCAD) падающего меню **Window** (Окно) в главном меню Visual LISP (см. рис. 1.24).

Допустим, вы находитесь в AutoCAD и хотите вернуться в среду Visual LISP. В таком случае введите команду **VLLIDE** (Вызов среды Visual LISP) в командной строке AutoCAD.

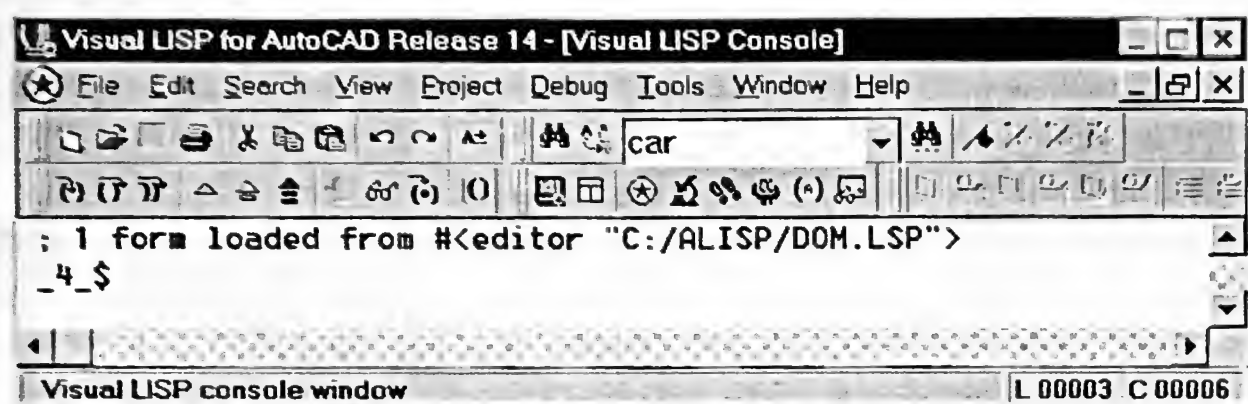


Рис. 1.34. Окно консоли Visual LISP

Окно консоли Visual LISP служит для отображения диагностических сообщений об ошибках в программах AutoLISP и результатах работы многих функций AutoLISP. Например, в окне консоли отображается результат работы функции (**PRINC...**).

Для отображения текущего значения переменной AutoLISP в Visual LISP достаточно набрать имя переменной в окне консоли и нажать клавишу **Enter** (Ввод).

Чтобы выполнить, оценить и просмотреть результаты, выражение AutoLISP можно вводить в окне консоли Visual LISP. Если требуется ввести больше одной строки, то для ввода следующей строки нажмите комбинацию клавиш **Ctrl+Enter**. Несколько выражений можно вводить до нажатия клавиши **Enter** (Ввод).

Помните, что у вас есть возможность копировать и передавать текст между консолью и окном текстового редактора. Большинство команд текстового редактора также доступны в окне консоли.

Нажатием клавиши **Tab** можно восстановить предыдущую команду, которая была введена в окне консоли. Неоднократное нажатие клавиши **Tab** восстанавливает более ранние команды. Если нажать комбинацию клавиш **Shift+Tab**, команды будут восстановлены в обратном направлении.

С помощью клавиши **Tab** можно осуществить ассоциативный поиск. Если, например, ввести выражение, которое начинается с символов (+, а затем нажать клавишу **Tab**, Visual LISP отыщет последнюю введенную команду, которая начинается с символов (+. Чтобы изменить направление поиска, следует нажать комбинацию клавиш **Shift+Tab**.

Клавиша **Esc** (**Escape** – отменить) удаляет любой текст после подсказки с консоли. Если нажать комбинацию клавиш **Shift+Esc**, можно будет отказаться от интерпретации текста, введенного с консоли в ответ на подсказку и перейти к следующей подсказке.

После нажатия правой кнопки мышки или комбинации клавиш **Shift+F10** в любом месте окна консоли на экране появляются меню команд

Visual LISP и опции. Они могут быть использованы для того, например, чтобы скопировать и вставить текст в командную строку консоли, найти текст или инициализировать Visual LISP.

1.5. Компилирование программ и выход из Visual LISP

При запуске программы AutoLISP с консоли Visual LISP или при загрузке программ из окна текстового редактора Visual LISP компилирует программу и выполняет ее.

Visual LISP имеет два компилятора: компилятор с непосредственным выполнением для быстрого тестирования и отладки новой программы (он подобен интерпретатору AutoLISP) и компилятор, который размещает откомпилированные программы с тем же именем, что и исходные, но с расширением FAS. Такие программы могут быть запущены из Visual LISP и выполняются быстрее, чем сгенерированные компилятором с непосредственным выполнением. Это позволяет создавать приложения, с которыми могут работать другие пользователи. Приложения могут быть автономные или сопровождаемые **Visual LISP Run-Time Support System (RTS)** (Система поддержки выполнения программ Visual LISP). Скомпилированные исполняемые файлы содержат только машинно-ориентированный код. Исходный текст зашифрован компилятором файла Visual LISP.

Исполняемые файлы, сгенерированные компилятором, часто называются «компилированными программами» или fas-файлами.

Visual LISP обеспечивает различные способы запуска и использования компилятора файла. Для того чтобы скомпилировать одиночный файл LISP, используется LSP-компилирующая функция.

Когда приложение состоит из набора файлов AutoLISP, загруженных одновременно, используйте интегрированные средства управления проектом Visual LISP для компилирования ваших файлов. Организатор проекта автоматически перетранслирует файлы, которые изменились, обеспечит поиск фрагментов программ даже в том случае, если вы не знаете, какой файл содержит их, и сможет оптимизировать использование функций, а также локальных переменных в компилируемых файлах.

Чтобы скомпилировать одиночный файл AutoLISP, вызовите функцию (**VLISP-COMPILE...**) в ответ на подсказку с консоли Visual LISP:

(VLISP-COMPILE <'режим> <"имя компилируемого файла"> <["имя скомпилированного выходного файла"]>)

<режим> определяет режим трансляции:

ST – стандартный режим;

LSM – режим, который оптимизирует, но не компоует;

LSA – режим, который оптимизирует и связывает.

Стандартный режим создает самый простой выходной файл и обычно применяется для программ, состоящих из единственного файла.

Режим с оптимизацией повышает эффективность компилируемых файлов и применяется в сложных и больших программах.

К базисным функциям оптимизации относятся:

- **Link Function** (Функция связи), которая вызывается, чтобы создать прямые ссылки к компилируемой функции в оттранслированной программе и защитить код против функционального переопределения во времени выполнения;
- **Drop Function** (Функция уменьшения), которая делает оттранслированную программу более безопасной, уменьшает размер программы и сокращает время загрузки.

Например, при компиляции программного файла PRIVET2W.LSP в Visual LISP на диске C в папке ALISP вы можете вызвать компилятор с консоли:

```
_S (VLISP-COMPILE 'ST "C:/ALISP/PRIVET2W.LSP ")
```

Если вы опускаете расширение файла с исходной программой, Visual LISP добавляет расширение LSP.

При определении пути в имени файла используйте символ наклонной черты вправо (/) или удваивайте наклонную черту влево (\\).

Выходной (скомпилированный) файл по умолчанию имеет то же имя, что и исходный, но расширение FAS. После завершения компиляции появится сообщение компилятора в окне **Build Output** (Создание выходного файла) (рис. 1.35).

Имя выходного файла можно изменять, например:

```
(VLISP-COMPILE 'ST "C:/PRIVETW.LSP " "C:/PRIVETF.FAS ")
```

В ходе трансляции по окончании каждой ее стадии компилятор печатает название функции и различные сообщения. Первая стадия трансляции – синтаксическая и лексическая проверка исходного текста. Если компилятор находит на этой стадии ошибки, он выдает сообщение и останавливает процесс трансляции. Когда компилятор встречается выражения, которые расцениваются как опасные, он выдает предупреждение. К опасным выражениям относятся, например, такие, которые переопределяют существующие функции AutoLISP или задают новые значения защищенным символам.



Рис. 1.35. Окно сообщений компилятора Build Output

Если трансляция завершается успешно, как в нашем примере на рис. 1.35, в окне **Build Output** (Создание выходного файла) появляется название скомпилированного выходного файла.

Может случиться, что трансляция завершилась с ошибками. В такой ситуации исходный текст можно просмотреть и отредактировать. Для этого необходимо дважды нажать на сообщение об ошибке в окне Build Output.

Чтобы загрузить и выполнить компилируемую программу, предстоит вначале загрузить ее, а затем запустить на выполнение.

Для загрузки программы необходимо ввести функцию (**LOAD...**) после подсказки с консоли Visual LISP.

```
_ $ (LOAD "C:/PRIVETW.FAS ")
```

Если вы не укажете расширение файла FAS, Visual LISP не забудет сделать это сам.

Загрузить программу можно также, если выбрать пункт Load File (Загрузить файл) падающего меню **File** (Файл) главного меню Visual LISP. На экране появится диалоговое окно **Load lisp file** (Загрузить lisp файлы). В поле **Тип файлов** (Files of type) обязательно выберите пункт **Compiled AutoLISP Files** (Скомпилированные файлы AutoLISP), иначе Visual LISP перечислит только LSP файлы.

Запустить программу на выполнение можно так же, как и программу, загруженную из окна текстового редактора. Для этого необходимо ввести с консоли после подсказки:

```
_ $ (PRIVET2W)
```

При разработке приложений, содержащих несколько файлов, необходимо использовать встроенную систему управления проектом Visual LISP для автоматической оптимизации кода.

Выход из среды Visual LISP можно осуществить двумя способами:

- с помощью меню. Для этого мышкой выберите пункт **Exit** (Выход) падающего меню **File** (Файл) главного меню;
- с помощью комбинации клавиш **Alt+Q**.

И в том и в другом случае на экране появится диалоговое окно **Exiting Visual LISP...** (Выход из Visual LISP) (см. рис. 1.14). На вопрос **Really want to quit session?** (Действительно ли вы хотите выйти из Visual LISP?) вы должны ответить **Yes** (Да) или **No** (Нет), то есть щелкнуть по соответствующей кнопке.

Если в тексте в любом окне редактора сделаны какие-либо изменения, и они не были сохранены, то во время закрытия файла появится диалоговое окно с предложением сохранить изменения (рис. 1.36).

В ответ на запрос **Save changes to DOM** (Сохранить изменения в файле под названием DOM) необходимо щелкнуть по кнопке **Yes** (Да), **No** (Нет) или **Cancel** (Отменить).

Если при выходе из Visual LISP остаются программы, которые открыты в окнах текстового редактора, Visual LISP автоматически открывает их в начале нового сеанса работы.

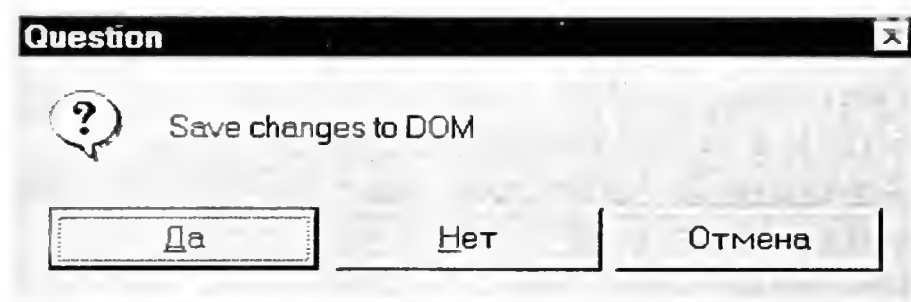


Рис. 1.36. Окно с запросом о сохранении изменений

ОСНОВЫ ПРОГРАММИРОВАНИЯ

Основные понятия и определения	48
Основные этапы программирования на AutoLISP	52
Встроенные функции языка AutoLISP	58
Дополнительные функции языка AutoLISP	72
Программирование на AutoLISP в среде Visual LISP ..	82

В этой главе поясняются основные понятия и определения языка AutoLISP, рассказывается об этапах программирования, встроенных функциях языка, а также функциях, расширяющих возможности AutoLISP. Здесь же описывается процесс разработки программ на языке AutoLISP в среде Visual LISP.

2.1. Основные понятия и определения

AutoLISP – один из диалектов языка программирования высокого уровня COMMON LISP (1986). LISP был создан как язык функционального программирования и относится к языкам декларативного типа.

Представленная на схеме классификация несколько условна, поскольку почти каждый язык содержит те или иные элементы другого языка. Со временем языки взаимно дополняют и обогащают друг друга.

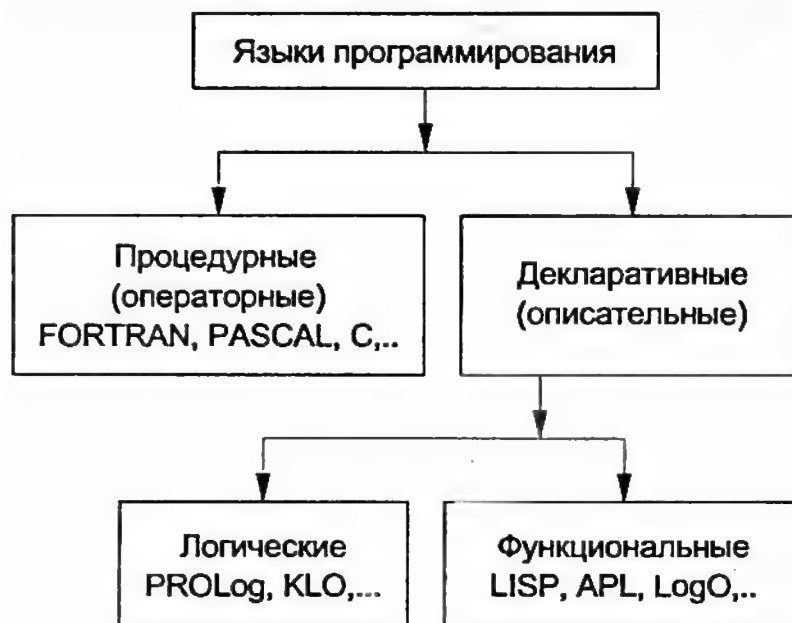


Рис. 2.1. Классификация языков программирования

Название языка LISP происходит от начальных букв двух слов **LISting Processing** (Обработка списков).

Список – это упорядоченная, заключенная в круглые скобки последовательность, элементами которой могут быть: числовые константы (числа целые и вещественные); текстовые константы; логические (**T** – истина, **NIL** – ложь); идентификаторы (имена переменной, функции, аргумента); список. Список будем записывать с апострофом перед открывающей круглой скобкой. Например:

' (15 12,14 "Текстовая константа" ' (A B C) PI)

Таким образом, список в общем виде – это многоуровневая, иерархическая структура данных, в которой открывающие и закрывающие круглые скобки находятся в строгом соответствии.

Список, в котором нет ни одного элемента, называется пустым и обозначается '()' или именем **NIL**. Список в AutoLISP заключается в круглые скобки, а элементы списка разделяются пробелами.

Выражение – это список, в котором первым элементом является имя функции. Любая функция AutoLISP состоит из выражений и сама является выражением. Выражение в AutoLISP имеет вид:

(<имя функции> [<аргумент1>] [< аргумент2>] [<>]...)

Квадратные скобки означают, что элемент может быть в списке, а может и не быть.

Аргументы (параметры) – это средство передачи значений (данных) в функцию. Аргументами могут быть переменные, константы (числовые, текстовые, логические) или выражения. Число аргументов функции может быть переменным, фиксированным или нулевым.

Переменные – это элементы языка AutoLISP, предназначенные для представления различных элементов языка AutoLISP и не имеющие постоянного значения.

Константы – это элементы языка AutoLISP, предназначенные для представления самих себя, имеющие постоянные значения.

Константы языка AutoLISP могут быть трех видов: числовые (целые и вещественные); текстовые (строковые) и логические.

Числовые константы – это любая последовательность цифр со знаком или без знака, отделяемая или неотделяемая точкой. Знак точки не должен стоять в начале и в конце числа.

Целые числа в AutoLISP представляются 32 двоичными разрядами со знаком, и их значения могут находиться в пределах $-2147483648... +2147483648$. Однако между AutoLISP и AutoCAD передаются только 16-разрядные числа (то есть значения в пределах $-32768... +32767$). При использовании значений, выходящих за эти пределы, необходимо использовать функцию (**FLOAT...**) для преобразования целого числа в вещественное.

Вещественные числа в AutoLISP представляются как числа с плавающей точкой с двойной точностью, при этом обеспечивается не менее 14 знаков точности. Вещественные числа передаются как значения с 32 двоичными разрядами. Числа между -1 и 1 должны явно содержать нулевую целую часть. Точно так же надо писать, например, 23.0, а не 23.

Текстовые константы – это любая последовательность знаков и пробелов, взятых в кавычки.

Логические константы – это Т (True – истина), Nil – ложь.

Символы (идентификаторы) – это имя (имена переменной, функции, аргумента), состоящее из прописных или строчных букв, цифр и знаков за исключением () – круглых скобок, . – точки, ' – апострофа, " – кавычек, ; – точки с запятой. Изначально у символов нет никакого значения.

В интерпретаторе прописные и строчные буквы отождествляются и представляются прописными.

Форма представления программы и обрабатываемых данных в AutoLISP одна и та же – список. В зависимости от контекста запись вида $(- 10 7)$ в AutoLISP может интерпретироваться двояко.

Это может быть список, обеспечивающий вычисление. Первый элемент списка — имя функции, два остальных — ее аргументы. В результате получаем значение вычитания, равное $10 - 7$, то есть 3. Но это может быть и список, состоящий из трех элементов (символов: —, 10 и 7).

Списками удобно представлять алгебраические выражения, графы, геометрические объекты, правила вывода и многие другие сложные объекты. Кроме того, они хорошо компонуется в памяти современных ЭВМ.

Единственный вид операций над данными в AutoLISP — это вычисление функций. Перед вычислением все переменные, аргументы и вызовы функций заменяются на свои значения. Чтобы обойти обязательное вычисление функций или блокировать их вычисление, используется функция (QUOTE...) или ее эквивалент в виде апострофа '. Например, запись

(QUOTE (* 5 6)) или '(* 5 6)

даст одно и то же значение (* 5 6) — список.

Как можно заметить, операция умножения не выполняется. Чтобы снять блокировку, используется функция (EVAL...); например, запись

(EVAL (QUOTE (* 5 6))) или (EVAL '(* 5 6))

даст одно и то же значение 30 — результат умножения чисел 5 и 6.

Функция (EVAL...) обеспечивает снятие блокировки и выполняет операцию вычисления, например, функции умножения. Все эти функции — встроенные функции языка AutoLISP.

Здесь функция (EVAL...) присутствует явно, однако при вычислении функций, не блокированных функцией (QUOTE...), нет ее явного присутствия, поэтому писать ее не надо. По существу, функция (EVAL...) — это интерпретатор языка AutoLISP (Evaluate — вычислить).

Работа языка AutoLISP состоит в чтении входного выражения (списка), вычислении этого выражения с помощью функции EVAL (она присутствует, как правило, неявно) и выводе полученного результата — значения функции. При этой схеме работы не происходит перевода программы в машинный код. Такой режим работы называется интерпретирующим, а функция (EVAL...) — интерпретатором.

Значением переменной в LISP может быть объект любого типа (число, символ, список, имя переменной, функции, ...) без предварительного его описания, что недопустимо в процедурных языках программирования.

Современные LISP-системы, включая AutoLISP, обладают мощными средствами, необходимыми для эффективного решения численных задач. Известно, что одна из целей COMMON LISP заключалась в том, чтобы

стать гибкой и эффективной альтернативой Фортрану в области численного научно-технического программирования.

В языке AutoLISP свыше 200 встроенных функций, каждая из которых выполняет определенную операцию. LISP является языком прикладного и системного программирования высокого уровня одновременно. Машинный язык он напоминает тем, что и данные, и программы представлены в нем в одинаковой форме.

В AutoLISP можно выделить три вида функций: встроенные в язык, разработанные и описанные пользователем, а также функции, разработанные и описанные другими пользователями.

Для создания новых функций в AutoLISP имеется специальная функция (**DEFUN...**). При разработке функций в AutoLISP широко используется мощный метод рекурсии, то есть при определении функции эта функция может вызывать саму себя.

Рассмотрим этот метод на примере вычисления показательной функции с целым положительным коэффициентом.

Показательную функцию можно представить как функцию двух аргументов: $Y(A, N) = A^N$. Эту же функцию можно представить и в другом виде:

$$Y(A, N) = A * Y(A, N-1) = A * A^{N-1} = A^N.$$

В языке AutoLISP эту функцию можно определить с помощью встроенной функции под именем **DEFUN** (**DE**fini**tion** **FUN**ction – определение функции) с использованием рекурсии:

```
(DEFUN РОС (A N) ; определение показательной функции
  (IF (= N 1) A
    (* A (РОС A (- N 1)))))
)
```

В результате выполнения функции (**DEFUN...**) появится имя РОС. Это означает, что создана и описана новая функция, которая может быть использована для вычислений.

Например, запись (РОС 5 4) тут же даст результат вычисления: 625 – возведение числа 5 в четвертую степень.

При создании пользовательской функции после имени функции **DEFUN** следует имя функции, которое задает сам пользователь (в нашем примере РОС), затем в круглых скобках через пробел указываются аргументы. Далее описывается алгоритм вычисления функции: если $N = 1$, то результат выполнения функции равен A , иначе произведению A на A^{N-1} . Здесь следует обратить особое внимание на форму записи функции при ее определении. Имя функции пишется в *инфиксной* форме (сначала имя функции, а затем в круглых скобках аргументы). При использовании функции для вычислений она пишется в *префиксной* форме. Имя функции и аргументы заключены в круглые скобки.

Функция должна описываться до начала использования. Обращение к описанной функции может быть помещено в любом месте программы, оно может стоять даже на месте аргумента другой функции.

При вызове функции интерпретатор обратится к описанию, произведет вычисление согласно описанию и тут же выдаст результат или будет использовать его для дальнейших вычислений. Обращение после описания нашей функции даст следующий результат:

```
(POC 5 (+ 1 3))
```

```
625
```

Для облегчения понимания описание функций на AutoLISP лучше делать с отступами, а также снабжать необходимыми комментариями, которые записываются после ; (точки с запятой).

```
(DEFUN POC (A N)                ; показательная функция с положительным
  (IF (= N 1) A                ; показателем степени
    (* A (POC A (- N 1)))))
)
```

```
(POC 5 4)                      ; вызов функции после ее описания
```

Функцию (DEFUN...) можно использовать для описания самых разнообразных функций, например, функции, которая будет брать производную от алгебраического выражения; искать и исправлять в тексте ошибки; рисовать на экране многоступенчатый объект с заданными параметрами; чертить узел машины...

В настоящее время созданы LISP-системы и даже LISP-машины, которые содержат тысячи подобных функций, обеспечивающих эффективное решение многочисленных самых разнообразных задач, включая задачи искусственного интеллекта.

2.2. Основные этапы программирования на AutoLISP

AutoLISP хорошо сочетается с прикладной графикой, он очень прост в изучении и универсален в применении. Язык имеет много встроенных функций, отражающих специфику графического редактора AutoCAD.

AutoLISP обеспечивает возможность формирования чертежа за считанные минуты. Чтобы получить чертеж, необходимо предварительно написать программу для его создания. Для упрощения процесса нужно иметь соответствующий банк программ графических чертежей.

Написание программы на AutoLISP требует выполнения ряда основных действий: четкой формулировки – (постановки) задачи; выявления

основных особенностей, взаимосвязей и количественных закономерностей; формирование графического изображения объекта; разработки программы (функций) на AutoLISP; реализации программы на ЭВМ.

Каждый из этих этапов рассмотрим подробнее.

Постановка задачи. На этом этапе определяется: графический объект (план офиса, коттедж, многоступенчатый объект и т. д.), который может формироваться программой при различной входной информации (например, число ступеней объекта, размеры каждой ступени, текстовые пояснения, необходимость указания размеров, местоположение объекта на чертеже и т. д.).

Выявление основных особенностей, взаимосвязей и количественных закономерностей. На этом этапе устанавливаются взаимосвязи отдельных элементов чертежа с точки зрения эффективного их построения, например, взаимосвязь между смежными ступенями объекта: диаметр последующей ступени объекта может быть меньше или больше диаметра предыдущей и т. д. Одновременно определяются способы построения отдельных элементов графического объекта и условия, которые влияют на выбор способа построения элемента. Проводится декомпозиция сложного графического объекта на ряд элементарных, описание которых может быть выполнено простыми встроенными функциями. Определяются постоянные и переменные параметры, текстовые вставки, формируются запросы и сообщения, программно выдаваемые конструктору и т. д.

Разработка последовательности действий создания графического изображения объекта. На этом этапе подробно излагаются все действия, связанные с построением графического изображения объекта, возможно использование циклических процедур. Указывается, какая входная информация необходима, как она используется для формирования изображения того или иного элемента объекта.

Разработка программы (функций) AutoLISP. В языке функционального программирования каждое действие, каждая операция описываются некоторой функцией или комбинацией функций, встроенных или построенных пользователями. Каждая функция имеет свое оригинальное имя, она может иметь один или несколько аргументов либо не иметь их. Аргументы функции могут быть тоже функциями.

В AutoLISP используется несколько необычная форма записи функции – *префиксная*. Все функции, встроенные и разработанные пользователями, имеют префиксную форму записи.

Префиксная форма записи функции – это такой вид записи, в которой имя функции вместе с аргументами функции заключаются в круглые скобки. Наиболее привычной является *инфиксная* форма записи функции.

Инфиксная форма записи функции – это такой вид записи, при которой сначала пишется имя функции, а в скобках указываются ее аргументы.

Ниже в общем виде приведены разные формы записи:

$F(X1, X2, \dots)$ – инфиксная форма записи функции;

$(F\ X1\ X2 \dots)$ – префиксная форма записи функции,

где F – имя функции;

$X1, X2$ – аргументы функции.

В качестве аргумента функции в AutoLISP может быть функция, переменная, константа и т. д. Имя функции может состоять из одного или нескольких символов. Например, запись $(+ 5 4)$ означает, что функция имеет имя $+$ (сложение) и два аргумента: числа 5 и 4. В результате выполнения функции получим число 9. Функция вычитания имеет имя $-$; функции деления $-$ /, умножения $- *$; возведение в степень – **EXPT**; определение модуля числа – **ABS**; синус – **SIN**; косинус – **COS**; арктангенс – **ATAN**; квадратный корень – **SQRT**; экспонента – **EXP**; натуральный логарифм – **LOG**.

Выполнение программы на AutoLISP. Рассмотрим основные способы выполнения программы на AutoLISP:

- *ввод программы с клавиатуры в командной строке системы AutoCAD.* В ответ на приглашение **Command:** можно вводить функции AutoLISP. После ввода функции в командной строке нажмите клавишу **Enter** (Ввод) для выполнения функций и просмотра результатов. Например:

```
Command: (+ (* 5 6 7) (COS 0.5)) ; нажмите клавишу Enter
210.878
```

```
Command: (COMMAND "LINE" "5,10" "30,40" "" "ZOOM" "ALL" "")
```

На графическом поле рабочего стола AutoCAD появится изображение линии, проведенной из точки с координатами (5, 10) в точку с координатами (30,40). Этот способ не предусматривает сохранения введенных выражений;

- *ввод программы, оформленной как описание функции с клавиатуры, в командной строке системы AutoCAD.* Если программа оформлена как одно или несколько описаний функций, то есть начинается с имени **DEFUN**, то к этой функции можно обращаться в любой момент до завершения текущего сеанса работы с AutoCAD, после чего программа должна быть набрана снова. Например, определим функцию, которая будет вычислять тангенс угла:

```
Command: (DEFUN TTANG (X) (IF (AND (/= X (* PI 0.5))
(/= X (* PI 1.5))) (/ (SIN X) (COS X)) "Бесконечность"))
TTANG
```

```
Command: (TTANG 0.5)
```

0.546302

Command: (TTANG (* PI 1.5))

"Бесконечность"

Если синтаксис при определении функции не нарушен, то после нажатия клавиши **Enter** на следующей строке появится имя новой определенной функции, которой можно пользоваться для расчетов;

- *запись программы в файле и ее вызов в командной строке системы AutoCAD.* Программа создается с помощью любого редактора текстов и сохраняется в файле с расширением LSP, например, DOM.LSP.

Для выполнения программы ее необходимо загрузить, используя функцию AutoLISP (LOAD...). Например:

Command: (LOAD "C:/ALISP/DOM")

Имя файла при этом задается в текстовом виде без расширения с указанием пути. Если загрузка завершена успешно, функция (LOAD...) возвращает имя последней функции, определенной в файле, если нет – имя файла (LOAD "<имя файла>");

- *ввод программы, оформленной как описание функции с начальными символами C:* с клавиатуры в командной строке системы AutoCAD. Если программа может быть оформлена как описание функции без аргументов, то есть начинается с имени DEFUN, а собственное имя функции имеет начальные символы C:, то к этой функции можно обращаться без круглых скобок в любой момент до завершения текущего сеанса работы с AutoCAD, после чего программа должна быть набрана снова;
- *автоматическая загрузка программы.* Если программа оформлена как описание функции без аргументов, т.е. начинается с имени DEFUN, а собственное имя функции имеет начальные символы C: и она загружена в файл ACAD.LSP, то такой файл автоматически загружается при запуске AutoCAD. Файл ACAD.LSP обычно содержит наиболее часто используемые функции. Такие функции вызываются на выполнение как команды AutoCAD без круглых скобок;
- *ввод программы из окна текстового редактора Visual LISP.* Когда программа находится в активном окне текстового редактора, щелкните по кнопке **Load activate edit window** (Загрузка активизированного окна редактирования) или выберите пункт **Load Text in Editor** (Загрузить текст из окна редактирования) из падающего меню **Tools** (Инструменты) главного меню Visual LISP. Как только на экране появится сообщение о загрузке программы в окне консоли Visual LISP, произведите обращение к функции и для ее выполнения нажмите клавишу **Enter**;
- *ввод программы из окна консоли Visual LISP.* После введения программы в окне консоли нажмите клавишу **Enter** (Ввод) для ее выполнения.

Программа на AutoLISP может обрабатывать данные из файлов, которые необходимо предварительно открыть с помощью функции (OPEN...). После работы все открытые файлы необходимо обязательно закрыть, используя функцию (CLOSE...).

Функция (OPEN...) часто используется в сочетании с функцией (SETQ...), что упрощает операцию ввода-вывода. Например, следующая запись:

```
(SETQ F1 (OPEN "C:/KUDR/VAL.DAT" "r" ))
```

означает, что на диске C: в каталоге KUDR производится поиск файла с именем VAL.DAT, который открывается для чтения "r" (read – чтение). Использование функции (SETQ...) упрощает операцию закрытия открытого файла, которая при этом выглядит следующим образом:

```
(CLOSE F1)
```

Кроме символа "r" (чтение), можно использовать символ "w" (write – запись), который означает, что файл данных открывается для записи. Если файл с таким именем уже существует, данные будут перезаписаны. Использование символа "a" (add – добавить) означает открытие файла данных для пополнения его новыми сведениями. Если такого файла нет, он создается.

Следует обратить особое внимание на необычную запись местонахождения файла. В ней используется символ / (прямая косая черта) вместо символа \ (обратной косой черты), если следовать правилам операционной системы MS DOS. Это вызвано тем, что символ \ в AutoLISP используется для ввода управляющих символов: \n – перевод строки; \r – возврат каретки; \t – табуляция. В то же время можно использовать и обратную косую черту \, но в следующей записи: \\. И тогда открытие файла можно записать так:

```
(SETQ F1 (OPEN "C:\\KUDR\\VAL.DAT" "r" ))
```

Для вывода на экран значения, на которое ссылается символ, необходимо набрать восклицательный знак и символ: ! <символ>.

Ввод в командной строке AutoCAD круглой открывающей скобки (означает подключение интерпретатора AutoLISP для обработки вводимого выражения. Количество круглых открывающих скобок в выражении, записанном на языке AutoLISP, должно точно соответствовать числу круглых закрывающих скобок.

Последняя парная закрывающая скобка означает конец ввода выражения на языке AutoLISP и переход интерпретатора AutoLISP в режим вычисления (обработки) выражения. Если введенное выражение полностью соответствует синтаксису языка AutoLISP, то ниже появляется результат вычисления.

Допустим, необходимо произвести сложение числа 5.34 с вещественным числом, которое вводится с клавиатуры. Это выражение можно записать в командной строке следующим образом:

```
(+ 5.34 (GETREAL "ВВЕДИТЕ ЧИСЛО N = "))
```

После нажатия клавиши **Enter** (Ввод) на экране появится предложение в виде:

```
ВВЕДИТЕ ЧИСЛО N =
```

Введем, например, число 15.06.

На следующей строчке будет показан результат вычисления:

```
20.4
```

Предположим, в выражение вкралась ошибка: пропущены вторые двойные кавычки в строковой константе в подсказке. Выражение будет выглядеть так:

```
(+ 5.34 (GETREAL "ВВЕДИТЕ ЧИСЛО N = ))
```

В этом случае после ввода выражения появится сообщение:

```
2>
```

Такое состояние позволяет продолжить ввод выражения AutoLISP.

Данное сообщение связано с тем, что все символы после первой открывающей кавычки воспринимаются как строка символов. Чтобы продолжить вычисления, необходимо в строке, где появилось сообщение 2>, закрыть строковую константу второй двойной кавычкой и ввести две закрывающие скобки.

Строка будет выглядеть следующим образом:

```
2> " ) )
```

После нажатия клавиши **Enter** появится запрос:

```
ВВЕДИТЕ ЧИСЛО N = ) )
```

Необходимо ввести число, например, 15.06.

```
ВВЕДИТЕ ЧИСЛО N = ) ) 15.06
```

```
20.4
```

Отказаться от неверно набранного выражения можно, если нажать комбинацию клавиш **Ctrl+C** и правильно набрать выражение.

Допустим, в выражении, приведенном выше, сделана ошибка – пропущена вторая закрывающая скобка:

```
(+ 5,34 (GETREAL "ВВЕДИТЕ ЧИСЛО N = ")
```

В этом случае после ввода выражения появится сообщение:

```
1>
```

Это означает, что в выражении пропущена одна закрывающая скобка.

Чтобы продолжить вычисления, необходимо в строке, где появилось сообщение 1>, ввести одну закрывающую скобку. Строка будет выглядеть так:

```
1> )
```

После нажатия клавиши **Enter** появится запрос:

```
ВВЕДИТЕ ЧИСЛО N =
```

Введем, например, число 15.06:

```
ВВЕДИТЕ ЧИСЛО N = 15.06
```

```
20.4
```

Отказаться от неверно набранного выражения также можно, если нажать комбинацию клавиш **Ctrl+C** и правильно набрать выражение.

Самое трудное в AutoLISP – отслеживать соответствие открывающих и закрывающих круглых скобок и кавычек для строковых констант. Чтобы облегчить этот процесс при написании программ на AutoLISP, целесообразно использовать текстовые редакторы, которые могут следить за тем, чтобы количество открывающих скобок соответствовало количеству закрывающих. Такими возможностями обладает, например, текстовый редактор NORTON Editor фирмы SYMANTEC. Но лучше всего использовать современную интегрированную среду Visual LISP для системы AutoCAD 14-й версии.

2.3. Встроенные функции языка AutoLISP

По назначению все встроенные функции языка AutoLISP условно можно разделить на следующие:

- ввода данных различного типа;
- манипулирования с данными;
- работы с числовыми данными;
- управления процессом вычисления (выполнения) функций;
- проверки выполнения условий;
- вывода данных различного типа;
- доступа к примитивам и средствам AutoCAD;
- особые.

Пользователь может переопределять или модифицировать функции, расширять и создавать свои собственные.

Ниже мы рассмотрим синтаксис основных встроенных функций языка AutoLISP, которые могут быть использованы в процессе разработки пользователем собственных функций.

2.3.1. Функции для ввода данных различного типа

(PROMPT <"Поясняющее сообщение ">) – ввод сообщения;

(PROMPT "ВВЕДИТЕ ОСНОВНЫЕ ДАННЫЕ ЗАДАЧИ ")

ВВЕДИТЕ ОСНОВНЫЕ ДАННЫЕ ЗАДАЧИ

(GETINT <"Поясняющее сообщение ">) – ввод целого числа;

(GETINT "\n ВВЕДИТЕ ЧИСЛО ЭТАПОВ РЕШЕНИЯ ")

ВВЕДИТЕ ЧИСЛО ЭТАПОВ РЕШЕНИЯ 3 ; ввод целого числа

(GETREAL <"Поясняющее сообщение ">) – ввод действительного числа;

(GETREAL "\n ПРОДОЛЖИТЕЛЬНОСТЬ ОПЕРАЦИИ ")

ПРОДОЛЖИТЕЛЬНОСТЬ ОПЕРАЦИИ 15.5 ; ввод действительного числа

(GETSTRING [<CR>][<"Поясняющее сообщение ">]) – ожидание ввода строки с пробелами, если есть <CR>, иначе ввод строки до 1-го пробела;

(GETSTRING "\n ВВЕДИТЕ ФАМИЛИЮ: ") ; вызов функции

ВВЕДИТЕ ФАМИЛИЮ: КУДРЯВЦЕВ ; ввод фамилии на запрос

(GETDIST [<T1>][<"Поясняющее сообщение ">]) – ожидание ввода точки T2, если есть T1, тогда расстояние $R = T2 - T1$, иначе вводится расстояние R – действительное или целое число;

(GETDIST '(1.0 1.0) "\n ВВЕДИТЕ ВТОРУЮ ТОЧКУ ")

ВВЕДИТЕ ВТОРУЮ ТОЧКУ 7,7 ; ввод координат второй точки

8.48528 ; результат - расстояние от

; точки (1,1) до (7,7)

Точки задаются списком из двух координат по X и Y, например, точке T1 может соответствовать список '(10.0 15.0);

(GETANGLE [<T1>][<"Поясняющее сообщение ">]) – ожидание ввода точки T2, если есть T1, тогда угол равен углу между отрезком [T1 T2] и осью X, иначе вводится значение угла в радианах;

(GETANGLE '(1.0 1.0) "\n УГОЛ НАКЛОНА ЛИНИИ ") ; вызов

УГОЛ НАКЛОНА ЛИНИИ 5,3 ; ввод координат

; второй точки

0.463648 ; результат - угол

; наклона линии

(GETPOINT [<T1>][<"Поясняющее сообщение ">]) – ожидание ввода списка значений координат X и Y точки, если нет T1, иначе рисует «резиную» линию от T1 до положения курсора;

(GETPOINT "ВВЕДИТЕ БАЗОВУЮ ТОЧКУ ВТ ") ; вызов функции

ВВЕДИТЕ БАЗОВУЮ ТОЧКУ ВТ 6,9 ; ввод точки на запрос

(GETCORNER <T1>[<"Поясняющее сообщение ">]) – изображение «резинового» прямоугольника от точки T1 до расположения курсора;

(GETKEYWORD [<"Поясняющее сообщение ">]) – ожидание ввода ключевого слова, описанного функцией (INITGET...);

(INITGET [<Сумма чисел>][<"ключевые слова...">]) – установка режима ввода данных. Сумма чисел реализует несколько режимов.

Числа:

- 1 – запрещен пустой ввод;
 - 2 – запрещен ввод нуля;
 - 4 – запрещен ввод отрицательных чисел;
 - 8 – пределы координат не контролируются;
 - 16 – ввод трехмерных точек;
 - 32 – используется пунктир для изображения «резиновых» линии или прямоугольника;
 - 64 – игнорируется координата Z трехмерной точки (только в **GETDIST** AutoCAD 11-й и 12-й версий);
 - 128 – возвращает произвольный код с клавиатуры;
- ключевые слова – определение списка ключевых слов через пробел;

(INITGET 1 "ДА НЕТ") ; вызов функции

(GETKEYWORD "ИЗМЕНИТЬ РЕЖИМ РАБОТЫ ? (ДА НЕТ) ")

ИЗМЕНИТЬ РЕЖИМ РАБОТЫ ? (ДА НЕТ) ДА ; ввод ключа

(READ-LINE) – ожидание ввода строки с клавиатуры;

(READ-LINE) ; вызов функции

ВНИМАТЕЛЬНО РАССТАВЛЯЙТЕ СКОБКИ ; ввод текста с клавиатуры

(READ-CHAR) – ожидание ввода символа с клавиатуры для преобразования его в числовой код ASCII;

(READ-CHAR) ; вызов функции

A ; ввод символа

65 ; результат выполнения - код ASCII.

(READ-LINE <F1>) – ввод записи из файла <F1>;

(READ-LINE (OPEN "C:\\CM\\CM1.TXT" "r"))

(READ-CHAR <F1>) – ввод символа из записи файла <F1>.

Файл <F1> предварительно должен быть открыт. Это делается следующим образом:

(SETQ F1 (OPEN "C:\\KUDR\\FILE.DAT" "r"))

Первая запись в кавычках – местоположение файла в каталоге, вторая – режим чтения (read). Кроме этого, могут быть режимы "w" (write – записи), "a" (add – добавления).

После открытия файла и использования его необходимо закрыть:

(CLOSE F1) ; закрытие файла <F1>

2.3.2. Функции для манипулирования с данными

(SETQ <переменная> <выражение>) – присвоение переменной значения выражения (атом, список, функция, ...);

```
(SETQ A 10           ; присвоение переменной A значения 10
      B "ABS"        ; присвоение B текстовой константы "ABS"
      C (COS 0.6))   ; присвоение C значения (COS 0.6))
```

Или

```
(SETQ A 10 B "ABS" C (COS 0.6)); вызов функции
```

(SET <'символ> <выражение>) – присвоение символу с апострофом значения выражения (атом, список, функция, ...);

```
(SET 'A 10)          ; присваивает символу 'A значение 10
10                   ; результат
```

(CAR <список>) – возвращение первого элемента списка;

```
(CAR '(B C D E F))   ; вызов функции
B                     ; результат - первый элемент списка
```

(CDR <список>) – возвращение списка без первого элемента;

```
(CDR '(B C D E F))   ; вызов функции
(C D E F)            ; результат - список без первого элемента
```

(CAAR <список>) – эквивалентно (CAR (CAR <список>));

(CDAR <список>) – эквивалентно (CDR (CAR <список>));

(CADR <список>) – эквивалентно (CAR (CDR <список>));

(CDDR <список>) – эквивалентно (CDR (CDR <список>));

(CADAR <список>) – эквивалентно (CAR (CDR (CAR <список>)));

(CADDR <список>) – эквивалентно (CAR (CDR (CDR <список>)));

и т. д. вплоть до четырех уровней вложенности;

(LAST <список>) – возвращение последнего элемента списка;

```
(LAST '(B C D E F))   ; вызов функции
F                     ; результат - последний элемент списка
```

(LIST <элемент> <элемент>...) – создание списка из элементов;

```
(LIST 'A 'B 'C 'D 'E) ; вызов функции
(A B C D E)           ; результат - список элементов
```

(APPEND <список>...) – соединение нескольких списков в один;

```
(APPEND '(A B) '(C D E)) ; вызов функции
(A B C D E)              ; результат - один общий список
```

(REVERSE <список>) – изменение порядка элементов на обратный;

```
(REVERSE '(A B C D E F)) ; вызов функции
(F E D C B A)            ; результат выполнения
```


(ASSOC <ключ> <список>) – поиск элемента в списке по ключу;

```
(SETQ LA (LIST '(1 A) '(2 B) '(3 C))) ; вызов функции
(ASSOC 2 LA) ; вызов функции
(2 B) ; подсписок с заданным ключом
```

(SUBST <новый элемент> <старый элемент> <список>) – замена в списке старого элемента на новый;

```
(SETQ LB '(A B C D E)) ; вызов функции
(SUBST 'Z 'B LB) ; вызов функции
(A Z C D E) ; список с замененным элементом
```

(MEMBER <элемент N> <список>) – выделение списка с элемента N;

```
(MEMBER 'D '(A B C D E F)) ; вызов функции
(D E F) ; результат выполнения
```

(NTH <число N> <список>) – выделение (N + 1)-го элемента списка, так как счет элементов в списке начинается с N = 0;

```
(NTH 3 '(A B C D)) ; вызов функции
D ; выделен 4-й элемент списка
```

(FIX <действ. число>) – преобразование действительного числа в целое;

```
(FIX 5.4) ; вызов функции
5 ; результат выполнения - целое число
```

(FLOAT <целое число>) – преобразование целого числа в действительное;

```
(FLOAT 5) ; вызов функции
5.0 ; результат выполнения -
; действительное число
```

(ANGTOS <угол в радианах> [<представление>][<точность>]) – преобразование угла в радианах в текстовую строку;

```
(ANGTOS 0.685) ; вызов функции
"39.248" ; угол в градусах как текстовая
; константа
```

(CHR <число>) – преобразование числа в символьный код ASCII;

```
(CHR 77) ; вызов функции
"М" ; результат - символьный код ASCII числа 77
```

(ASCII <"символ">) – значение символа в числовом коде ASCII;

```
(ASCII "G") ; вызов функции
71 ; результат - числовой код ASCII символа G
```

(ATOF <"число">) – преобразование числовой текстовой константы в действительное число;

```
(ATOF "55.2") ; вызов функции
55.20000 ; результат выполнения
```

(atoi <"число">) – преобразование числовой текстовой константы в целое число;

```
(atoi "34")      ; вызов функции
34                ; результат выполнения
```

(itoa <целое число>) – преобразование целого числа в числовую текстовую константу;

```
(itoa 19)          ; вызов функции
"19"              ; результат выполнения
```

(rtos <действительное число>)[<режим>][<точность>]) – преобразование действительного числа в текстовую константу.

```
(rtos 34.5 1 4) преобразует 34,5 в "3.4500E+01" - научный режим
(rtos 34.5 2 4) преобразует 34,5 в "34.5000"    - десятичный режим
(rtos 34.5 3 4) преобразует 34,5 в "2'-10.5000"  - технический режим
(rtos 34.5 4 4) преобразует 34,5 в "2'-10 1/2"   - архитектурный режим
(rtos 34.5 5 4) преобразует 34,5 в "34 1/2"     - дробный режим
```

(strlen <"строка">) – определение числа символов в строке;

```
(strlen "AutoLISP") ; вызов функции
8                  ; результат - число символов в строке
```

(strcat <"строка 1"> <"строка 2">...) – соединение строк;

```
(strcat "Auto" "CAD") ; вызов функции
"AutoCAD"             ; результат соединения строк
```

(substr <"строка"> <начало> [<длина>]) – выделение части строки подстроки (начало – номер символа, длина – число символов);

```
(substr "AutoCAD" 5) ; вызов функции
"CAD"               ; результат - выделена часть текста
```

(length <список>) – определение длины списка;

```
(length '(A B C D E F)) ; вызов функции
6                        ; результат - число элементов
```

2.3.3. Функции для работы с числовыми данными и выражениями

(+ <N1> <N2> <N3>...) – сложение чисел N1, N2, N3, ...;

```
(+ 12.3 45 3.6 1) ; вызов функции
61.9              ; результат сложения нескольких чисел
```

(- <N1> <N2> <N3>...) – вычитание из числа N1 чисел N2, N3, ...;

```
(- 50.5 45.1 3 2.3) ; вызов функции
0.1                 ; результат вычитания нескольких чисел
```

(* <N1> <N2> <N3> ...) – перемножение чисел N1, N2, N3, ...;

(* 3.3 4 5 2)

; вызов функции

132.0

; результат умножения нескольких чисел

(/ <N1> <N2> <N3> ...) – деление числа N1 на N2, N3, ...;

(/ 70 5.5 2)

; вызов функции

6.36364

; результат деления

(ABS <N1>) – определение абсолютного значения числа N1;

(ABS -55.4)

; вызов функции

55.4

; результат определения абсолютного числа

(1+ <N1>) – добавление к числу N1 единицы;

(1+ 15)

; вызов функции

16

; результат выполнения

(1- <N1>) – вычитание из числа N1 единицы;

(1- 43)

; вызов функции

42

; результат выполнения

(SIN <угол в радианах>) – вычисление синуса угла в радианах;

(SIN 0.54)

; вызов функции

0.514136

; результат вычисления синуса

(COS <угол в радианах>) – вычисление косинуса угла в радианах;

(COS 1.4)

; вызов функции

0.169967

; результат вычисления косинуса

(ATAN <N>) – определение угла в радианах (арктангенс);

(ATAN 45)

; вызов функции

1.54858

; результат вычисления арктангенса

(EXP <N>) – определение величины экспоненты;

(EXP 2)

; вызов функции

7.38906

; результат вычисления экспоненты

(EXPT <N1> <N2>) – определение показательной функции;

(EXPT 6.5 2.3)

; вызов функции

74.08

; результат вычисления

(LOG <N>) – определение величины натурального логарифма;

(LOG 68.7)

; вызов функции

4.22975

; результат вычисления

(MAX <N1> <N2> ...) – поиск максимального числа из всех;

(MAX 3 9 5.5 4.5)

; вызов функции

9.0

; результат – максимальное число из всех

(MIN <N1> <N2>...) – поиск минимального числа из всех;

(MIN 5 4.1 2.2 7) ; вызов функции
2.2 ; результат – минимальное число из всех

(REM <N1> <N2>) – определение остатка от деления N1 на N2;

(REM 54 10) ; вызов функции
4 ; результат – остаток от деления

(GCD <N1> <N2>) – определение наибольшего общего делителя;

(GCD 12 27) ; вызов функции
3 ; результат – наибольший общий делитель

(ANGLE <T1> <T2>) – определение угла в радианах между прямой, проходящей через T1 и T2, и осью X;

(ANGLE '(2.0 3.0) '(8.0 9.0))
0.785398 ; результат вычисления – угол в радианах

(DISTANCE <T1> <T2>) – расстояние между точками T1 и T2;

(DISTANCE '(8.0 3.0) '(2.0 6.0)) ; вызов функции
6.7082 ; результат – расстояние между
; точками T1 и T2

(INTERS <T1> <T2> <T3> <T4> [<ON>]) – определение точки пересечения двух отрезков, проходящих соответственно через точки T1, T2 и T3, T4. Если <ON> нет, то точка пересечения должна быть в пределах обоих отрезков, иначе отрезки воспринимаются как бесконечные прямые;

(INTERS '(1 1) '(5 5) '(1 5) '(5 1)) ; вызов функции
(3.0 3.0) ; результат – точка
; пересечения отрезков

(POLAR <T1> <U> <R>) – определение точки T2 путем перемещения из точки <T1> под углом <U> в радианах на расстояние <R>;

(POLAR '(1 1) (/ PI 4) 4) ; вызов функции
(3.82843 3.82843) ; результат – координаты точки

2.3.4. Функции управления процессом вычисления (выполнения) функций

(COND (<(условие 1)> <(функция 1,1)> [(функция 1,2)]...)

.....
(<(условие N)> <(функция N,1)> [(функция N,2)]...) – вычисление тех функций, для которых выполняются условия;

Проверка проводится для всех условий.

(SETQ I 5 J 9) ; присваивание значений переменным I и J
(COND ((= I J) (MAX 5 10 3))) ; определение максимального числа

((< I J) (MIN 5 6 20)) ; определение минимального числа

((> I J) (* 5 7 78 9))) ; определение произведения чисел

5 ; результат выполнения (вычисления)

(IF (<(условие 1)> <(функция 1)> [(функция 2)]) – если (условие 1) выполняется, то вычисляется (функция 1), иначе (функция 2);

(SETQ I 5)

(IF (= I 0) (MAX 5 10 3) ; определение максимального числа

(MIN 5 6 20)) ; определение минимального числа

5 ; результат выполнения (вычисления)

(REPEAT <целое число> <выражение>...) – вычисление выражений заданное число раз;

(SETQ D 5)

(REPEAT 3 ; формирует цикл

(SETQ D (* D 2))) ; присваиваются D результаты расчета

40 ; результат расчета

(PROGN <выражение>...) – вычисление нескольких выражений там, где допускается вычисление только одного;

(IF (< C D) (PROGN (SETQ C (* C 2))

(SETQ D (* D 5))))

(APPLY <'имя функции> <'(список аргументов)>) – применение заданной функции к списку аргументов;

(APPLY '+ '(5 10.53 3)) ; вызов функции

18.53 ; результат сложения элементов списка

(MAPCAR <'имя функции> <'(список аргументов)>...) – выполнение заданной функции над первыми элементами списков, затем вторыми и т. д.;

(MAPCAR '* '(5 10 3) '(6 7 8) '(1.5 3 5)) ; вызов функции

(45.0 210 120) ; результат вычисления

(FOREACH <переменная> <список> <(функция 1)> <(функция 2)>...) присваивание переменной значения первого элемента списка и вычисление функций, затем второго и т. д.;

(FOREACH N '(0.5 1.0 1.5)

(PRINT (COS N))) ; вычисление COS для каждого

0.877583 ; элемента списка и печать

0.540302 ; результатов расчета

0.070737

(WHILE <(условие)> <(функция 1)> <(функция 2)>...) – вычисление в цикле функций до тех пор, пока выполняется условие;

(SETQ N 1) ; определение начального значения N

(WHILE (< N 4) ; условие вычисления в цикле

```
(PRINT (SIN (* 0.2 N))) ; вычисление SIN, печать результата
(SETQ N (+ N 1)) ; определение нового значения N
0.198669 ; результаты расчета
0.389418
0.564642
```

2.3.5. Функции проверки выполнения условий

(AND <выражение>...) – Т (TRUE), если нет NIL, иначе NIL;

```
(AND 10 15.5 "ABC") ; вызов функции
T ; результат выполнения
```

(OR <выражение>...) – Т, если хотя бы одно Т, иначе NIL;

```
(OR NIL "CAD" 15) ; вызов функции
T ; результат выполнения
```

(NOT <выражение>...) – Т, если имеется NIL, иначе NIL;

```
(SETQ A 15.8)
(NOT A) ; вызов функции
NIL ; результат выполнения
```

(BOUND P <атом>) – Т, если <атом> имеет значение, иначе NIL;

```
(SETQ B 12.3)
(BOUND P 'B) ; вызов функции
T ; результат выполнения
```

(NUMBERP <элемент>) – Т, если <элемент> число, иначе NIL;

```
(NUMBERP 183.2) ; вызов функции
T ; результат выполнения
```

(MINUSP <число>) – Т, если <число> отрицательное, иначе NIL;

```
(MINUSP 9.25) ; вызов функции
NIL ; результат выполнения
```

(EQUAL <выражение1> <выражение2> [<допуск>]) – Т, если оба выражения равны между собой, иначе NIL;

```
(EQUAL "ATOM1" "ATOM1" ) ; вызов функции
T ; результат выполнения
```

(LISTP <элемент>) – Т, если <элемент> список, иначе NIL;

```
(LISTP '(D E F)) ; вызов функции
T ; результат выполнения
```

(ATOM <элемент>) – Т, если <элемент> атом, иначе NIL;

```
(SETQ C '(A B D))
(ATOM C) ; вызов функции
NIL ; результат выполнения
```

(EQ <выражение1> <выражение2>) – T, если идентичны, иначе NIL;
 (SETQ E1 '(X Y Z) E2 '(X Y Z))

(EQ E1 E2) ; вызов функции

NIL ; результат выполнения

(/= <атом1> <атом2>...) – T, если <атом1> не равен остальным атомам, иначе NIL;

(/= 7.85 7.74) ; вызов функции

T ; результат выполнения

(< <атом1> <атом2>...) – T, если <атом1> меньше всех последующих, иначе NIL;

(< "A" "D") ; вызов функции

T ; результат выполнения

(<= <атом1> <атом2>...) – T, если <атом1> меньше или равен всем последующим, иначе NIL;

(= <атом1> <атом2>...) – T, если <атом1> равен всем остальным атомам, иначе NIL.

2.3.6. Функции для вывода данных различного типа

(WRITE-CHAR <целое число>) – печатание символа ASCII кода;

(WRITE-CHAR 67) ; после ввода появляется эхо команды,
 67 ; а на экран выводится латинская буква C

(WRITE-LINE <текст>) – печатание текста без кавычек;

(PRIN1 <выражение>) – печатание и возвращение значения <выражения>; если <выражение> – текст, то в кавычках;

(PRIN1 "HELLOW ") "HELLOW "

(PRINC <выражение>) – печатание и возвращение значения <выражения>; если это текст, то без кавычек;

(PRINT <выражение>) – печатание с новой строки с последующим пробелом и возвращение значения выражения, текст без кавычек;

(PRINT "HELLOW") ; вызов функции

HELLOW ; результат выполнения

(WRITE-CHAR <целое число> <F1>) – печатание текста в файл без кавычек, а возвращение в кавычках;

(WRITE-LINE <текст> <F1>) – печатание текста в файл <F1> без кавычек, а возвращение в кавычках;

(SETQ F1 (OPEN "C:\\AUTOCAD\\LISP\\WR.LSP" "a"))

(WRITE-LINE "ХОРОШЕЕ НАЧАЛО" F1)

(**PRIN1** <выражение> <F1>) – печатание <выражения> в файл <F1> и возвращение значения <выражения>, текст – в кавычках;

(**PRINC** <выражение> <F1>) – печатание в файл <F1> и возвращение значения <выражения>, текст без кавычек;

```
(PRINC "ПРИВЕТ ВСЕМ ПОЛЬЗОВАТЕЛЯМ" F1 )
```

(**PRINT** <выражение> <F1>) – печатание в файл <F1> и возвращение значения выражения, текст без кавычек.

2.3.7. Функции доступа к примитивам и средствам AutoCAD

(**ENTGET** <имя примитива>) – выбирает примитив из базы данных и возвращает его в виде списка в кодах DXF;

(**ENTNEXT** <имя примитива>) – выбирает следующий примитив из базы данных;

(**ENTLAST**) – выбирает последний созданный примитив (объект) из базы данных.

Допустим, что последним мы ввели текстовый объект с такими данными:

```
Command: TEXT
```

```
JUSTIFY/STYLE/<START POINT>: 1,4
```

```
HEIGHT <0.2000>: 1.5
```

```
ROTATION ANGLE <0>: Enter
```

```
TEXT: Доброе утро
```

```
TEXT: Enter
```

После вызова функций (**ENTLAST**) и (**ENTGET...**) в командной строке AutoCAD получим представление последнего текстового объекта в виде списка DXF.

```
Command: (SETQ E (ENTGET(ENTLAST)))
```

```
((-1 . <ENTITY NAME: 31D0538>) ; имя примитива
(0 . "TEXT") ; тип объекта (примитива)
(5 . "4F") ; метка
(100 . "ACDBENTITY")
(67 . 0) (8 . "0")
(100 . "ACDBTEXT")
(10 1.0 4.0 0.0) ; начальная точка
(40 . 1.5) ; высота
(1 . "Доброе утро") ; текст
(50 . 0.0) ; угол поворота (радианы)
(41 . 1.0) ; степень растяжения
(51 . 0.0) ; угол наклона
```



```

(7 . "STANDARD")           ; гарнитура шрифта
(71 . 0)                    ; флаги генерации
(72 . 0)                    ; тип выравнивания
(11 0.0 0.0 0.0)           ; точка выравнивания
(210 0.0 0.0 1.0)          ; вектор направления выдавливания
(100 . "ACDSTEXT")
(73 . 0)
)

```

(ENTSEL [<подсказка>]) – вызывает имя примитива по указанной точке;

Command: (SETQ E (ENTSEL "Пожалуйста, выберите примитив: "))

Пожалуйста, выберите примитив: 1,4

(<ENTITY NAME: 31D0538> (1.0 4.0 0.0))

(HANDENT <метка>) – возвращает имя примитива по метке;

Command: (HANDENT "4F")

<ENTITY NAME: 31D0538>

(TEXTBOX...) – возвращает диагональные координаты поля, которое включает текстовый объект. Если использовать предыдущий пример, то результат от вызова функции (TEXTBOX E) будет таким:

Command: (TEXTBOX E)

((0.0 -0.25 0.0) (16.5 1.5 0.0))

(SSGET [<способ выбора примитивов>][<точка1>[<точка2>]]) – создает набор примитивов:

```

(SSGET)                     ; запрашивает способ выбора примитивов
(SSGET "T")                 ; выбирает текущий набор примитивов
(SSGET "П")                 ; выбирает последний примитив
(SSGET '(2 3))              ; выбирает примитив, проходящий через точку
(SSGET "P" '(0 0) '(3 4))   ; выбирает примитив в рамке
(SSGET "C" '(0 0) '(3 4))   ; выбирает примитивы, пересекаемые рамкой
(SSGET "X" <фильтр-список>) ; выбирает примитивы в соответствии
                           ; с фильтром-списком

```

Фильтр-список – это список точечных пар подобных типу списка, возвращаемых функцией (ENTGET...).

Например:

(SSGET "X" '((0 . "CIRCLE"))) ; возвращает набор из кругов

Для функции (SSGET "X"...) доступны следующие коды:

- 0 – тип примитива;
- 2 – имя блока для описания блока;
- 6 – имя типа линии;
- 7 – имя гарнитуры шрифта текста или атрибутов;
- 8 – имя слоя;

- 38 – уровень;
 39 – высота;
 62 – код цвета (0 по блоку и 256 – по слою);
 66 – следующий за атрибутом флаг в описании блока;
 210 – вектор направления выдавливания.
 (SSLENGTH <набор>) – определяет число примитивов в наборе;
 (SSNAME <набор> <номер примитива в наборе>) – возвращает имя примитива по номеру примитива в наборе. Первый примитив имеет номер 0;
 (SSADD [<имя примитива>][<набор>]) – добавляет имя примитива в набор;
 (SSDEL [<имя примитива>][<набор>]) – удаляет имя примитива из набора;
 (SSMEMB [<имя примитива>][<набор>]) – проверяет наличие примитива в наборе.

2.3.8. Особые функции

(QUOTE <выражение>) – блокировка вычисления выражения:

(QUOTE (A B C D)) ; вызов функции
 (A B C D) ; результат выполнения функции
 ' <выражение> – блокировка вычисления выражения;
 ' (A B C D E) ; вызов функции
 (A B C D E) ; результат выполнения

(EVAL <выражение>) – вычисление любого, правильно составленного выражения (обычно присутствует неявно).

Например, выражения (EVAL (ABS -3.5)) и (ABS -3.5) равносильны;

(DEFUN <имя функции> ([<список аргументов>...] [/<локальная переменная>]) <выражение>...) – определение новой функции: имя функции – это любая последовательность символов; список аргументов – это список переменных, значения которых должны быть определены до выполнения функции; локальные переменные – это переменные, действие которых ограничено данной функцией. Значением функции является значение последнего выполняемого выражения. Память под локальные переменные выделяется только во время вычисления функции;

(DEFUN SIN2 (X) ; функция под именем SIN2
 (* (SIN X) (SIN X))) ; вычисляет SIN в квадрате

(LAMBDA <аргумент> <выражение>...) – определение функции без имени;

(APPLY ' (LAMBDA (X Y) (+ (* X X) (* Y Y))) '(5 3))

34 ; результат выполнения функции без имени

(FINDFILE <"имя файла">) – поиск файла с данным именем, а возвращение и с именем каталога;

```
(FINDFILE "VAL1.LSP")      ; вызов функции
"C:\\ALISP\\VAL1.LSP"      ; результат выполнения
```

(LOAD <"имя файла">) – функция загрузки файла и выполнения его, имя файла без расширения (подразумевается расширение LSP);

```
(LOAD "C:\\ALISP\\VAL1")
```

(GETVAR <"имя системной переменной">) – определение значения системной переменной AutoCAD в данный момент;

```
(GETVAR "DIMTXT" )        ; определение высоты символов текста
0.18                      ; результат
```

(SETVAR <"имя системной переменной"> <значение>) – присваивание системной переменной AutoCAD определенного значения;

```
(SETVAR "BLIPMODE" 0)     ; удаление изображения маркера
(SETVAR "CMDECHO" 0)      ; удаление эха команд
```

(TRACE <имя функции>...) – выдача результатов выполнения функций;

(UNTRACE <имя функции>...) – конец отслеживания функции.

2.4. Дополнительные функции языка AutoLISP

(AUTOCAD_COLORDLG <целое число> [NIL]) – определение цвета.

Целое число в диапазоне 0–256 определяет цвет;

```
(AUTOCAD_COLORDLG 3)      ; определяет зеленый цвет
```

(AUTOCAD_HELPDLG HELPFILE TOPIC) – вызывает устаревшие средства справки;

(AUTOCAD_STRLSORT <список>) – сортирует список строк в алфавитном порядке;

```
(SETQ MOS ' ( "JAN" "FEB" "MAR" "APR" "MAY" "JUN" "JUL"
              "AUG" "SEP" "OCT" "NOV" "DEC" ))
```

```
(AUTOCAD_STRLSORT MOS)
```

```
("APR" "AUG" "DEC" "FEB" "JAN" "JUL" "JUN" "MAR" "MAY" "NOV" "OCT" "SEP")
```

(ACTION_TILE <имя подокна> <выражение действия>) – определяет действия, когда пользователь выбирает определенное подокно в диалоговом окне;

(ADD_LIST <строка>) – добавляет или изменяет строки в настоящее время в активном списке диалогового окна;

```
(SETQ LLIST ' ("первая строка " " вторая строка " " третья строка " ))
(START_LIST "LONGLIST")
```

```
(MAPCAR 'ADD_LIST LLIST)
(END_LIST)
```

(ADS) – возвращает список имен в настоящее время загруженных приложений и их пути;

```
(ADS) ; вызов функции
("FILES/PROGS/PROG1" "PROG2") ; результат выполнения
```

(ALERT <строка>) – отображает поле ALERT с ошибкой или предупреждающим сообщением, переданным как строка.

Поле ALERT – диалоговое окно с одиночной кнопкой **OK**;

```
(ALERT " Эта функция недоступна. ")
```

(ALLOC <целое число>) – устанавливает размер сегмента к данному числу узлов. Функция возвращает предыдущий размер сегмента;

(ARX) – возвращает список в настоящее время загруженных приложений ARX с их путями;

```
(ARX)
("FILES/PROGS/PROG1" "PROG2")
```

(ARXLOAD <приложение> [параметр]) – загружает приложение ARX;

```
(ARXLOAD "/MYAPPS/APPX")
"/MYAPPS/APPX"
```

(ARXUNLOAD <приложение> [параметр]) – разгружает приложение ARX;

```
(ARXUNLOAD "APPX")
"APPX"
```

(ATOMS-FAMILY <0 или 1> [список строк]) – возвращает список в настоящее время определенных символов: если 0, то совокупность атомов возвращает имя символа как список. Если 1, возвращается имя символа как список строк;

```
(ATOMS-FAMILY 1 ' ("CAR" "CDR" "XYZ" ))
("CAR" "CDR" NIL )
```

(AUTOARXLOAD <имя> <список строк>) – предопределяет название команды, чтобы загрузить связанный файл ARX;

```
(AUTOARXLOAD "BONUSAPP" ' (" APP1 " " APP2 " " APP3 " ))
```

(AUTOLOAD <имя LSP файла> <список строк>) – загружает LSP файл;

```
(AUTOLOAD "BONUSAPP" ' (" APP1 " " APP2 " " APP3 " ))
```

(AUTOXLOAD <имя приложение ADS> <список строк>) – предопределяет название команды, чтобы загрузить связанное приложение ADS;

```
(AUTOXLOAD "BONUSAPP" ' (" APP1 " " APP2 " " APP3 " ))
```


(**CLIENT_DATA_TILE** <подокно> <данные>) – связывает управляемые приложением данные с подокном диалогового окна;

(**DICTADD** <имя словаря> <имя объекта> <объект>) – добавляет графический объект в словарь с уникальным именем объекта;

(**DICTNEXT** <имя объекта> [S]) – находит имя объекта в словаре. Если параметр S есть и не NIL, то восстанавливается поиск сначала. Если вход найден, возвращается список точечных пар кодов типа DXF и значений;

(**DICTREMOVE** <имя словаря> <имя объекта>) – удаляет объект из словаря;

(**DICTRENAME** <имя словаря> <старое имя> <новое имя>) – переименовывает старое имя на новое в словаре;

(**DIMX_TILE** <"подокно">) – возвращает ширину подокна;

(**DIMY_TILE** <"подокно">) – возвращает высоту подокна.

Функции (**DIMX_TILE**...) и (**DIMY_TILE**...) используются с функциями (**VECTOR_IMAGE**...), (**FILL_IMAGE**...) и (**SLIDE_IMAGE**...), которые требуют, чтобы были определены абсолютные координаты подокна;

(**DONE_DIALOG** [целое число]) – завершает диалоговое окно и выдает координаты диалогового окна. Эта функция вызывается изнутри выражения действия или функции повторного вызова. 0 – отмена диалогового окна. Значение состояния, большее чем 1, зависит от вашего приложения;

(**END_IMAGE**) – окончание создания активного изображения диалогового окна. Эта функция – дополнение к функции (**START_IMAGE**...);

(**END_LIST**) – окончание обработки активного списка диалогового окна. Эта функция – дополнение к функции (**START_LIST**...);

(**COMMAND** [команды AutoCAD и их параметры]...) – выполняет заданные команды AutoCAD;

Строка пустого указателя (" ") эквивалентна нажатию клавиши **Enter** на клавиатуре. Вызов функции (**COMMAND**...) без параметров эквивалентен нажатию клавиши **Esc** и отменяет большинство команд AutoCAD.

Функция (**COMMAND**...) оценивает каждый аргумент и посылает его в AutoCAD в ответ на последовательные подсказки.

```
(SETQ PT1 '(2 3) PT2 '(7 8))
```

```
(COMMAND "LINE" PT1 PT2 "")
```

В результате будет изображена линия от точки с координатами (2,3) до точки (7,8).

Если приложение использует иностранный язык в AutoCAD, названия команд должны начинаться с подчеркивания, с тем чтобы они могли транслироваться. Когда используется префикс точки (чтобы избежать использования переопределенных команд), подчеркивание можно размещать в любом порядке: и **"_LINE"**, и **"_.LINE"** допустимы.

Команды, выполненные из функции (COMMAND...), не отображаются на экране в командной строке, если переменная системы CMDECHO установлена в 0.

Функции ввода пользователя (GETXXX...), (GETANGLE...), (GETSTRING...), (GETINT...), (GETPOINT...) и т. д. не должны использоваться внутри функции (COMMAND...). В противном случае на экране появится следующее сообщение: **AutoCAD rejected function** (AutoCAD отклонил функцию).

(CVUNIT <число или список чисел> <исходная система измерения> <требуемая система измерения>) – преобразует числа из одной системы измерения в другую.

Функция (CVUNIT...) успешно проводит преобразование, если используемые системы измерений есть в файле AUTOCAD.UNT и если они совместимы по размерности;

```
(CVUNIT 3 "MINUTE" "SECOND")      ; вызов функции
180.0                               ; результат
(CVUNIT 2.0 "INCH" "CM")
5.08
(CVUNIT '(1.0 3.4) "FT" "CM")
(30.48 103.632)
```

(GETENV...) – обеспечивает доступ подпрограммам AutoLISP к текущим значениям переменных среды операционной системы;

(GETCFG...) и (SETCFG...) – позволяют в приложениях AutoLISP просматривать и изменять значения параметров в разделе APPDATA файла AUTOCAD14.CFG.

(MENUCMD...) – управляет показом графических окон меню. Эта функция отображает, изменяет или делает запрос с одного из подменю текущего меню и принимает строковый параметр, который определяет подменю и его действие, чтобы впоследствии выполнить его.

Строковый параметр состоит из двух полей, отделяемых знаком =. Он может выглядеть, к примеру, следующим образом:

```
"MENU_AREA=ACTION"
(MENUCMD "S=OSNAP")      ; отображает экранное подменю ** OSNAP
```

(TEXTSCR) – отображает текстовое окно;

(TEXTPAGE) – очищает текстовое окно перед отображением текстового окна;

(REDRAW) – перерисовка всей графической области, она может определять одиночный объект, который будет (или нет) выведен повторно. Если объект сложный, типа ломаной линии или блока, может быть произведена перерисовка всего объекта или его заголовка;

(**GRTEXT...**) – обеспечивает показ текста непосредственно на экране или в областях меню, с высвечиванием или без него;

(**GRDRAW...**) – рисует вектор в текущей области просмотра с контролем над цветом и высвечиванием;

(**GRVECS...**) – рисует множественные векторы.

2.4.1. Примеры определения пользовательских функций

Определим функцию под именем **COPYL**, которая строит копию списка **L**.

```
(DEFUN COPYL (L)
  (COND ((NULL L) NIL)
        (T (CONS (CAR L) (COPYL (CDR L))))))
)
(COPYL '(A B C D E))      ; обращение к функции
(A B C D E)               ; результат
```

Определим функцию под именем **REMOVE**, которая удаляет из списка **L** все совпадающие с данным символом **A** (списком) элементы.

```
(DEFUN REMOVE (A L)
  (COND ((NULL L) NIL)
        ((EQUAL A (CAR L)) (REMOVE A (CDR L)))
        (T (CONS (CAR L) (REMOVE A (CDR L))))))
)
(REMOVE 'A '(C D E A G))  ; обращение к функции
(C D E G)                 ; результат
```

Определим функцию под именем **EARLYAB**, которая проверяет, находится ли элемент **A** ранее элемента **B** в списке **L** в соответствии с заданным упорядоченным списком **LU**.

```
(DEFUN EARLYAB (A B LU)
  (COND ((NULL LU) NIL)
        ((EQ A (CAR LU)) T)
        ((EQ B (CAR LU)) NIL)
        (T (EARLYAB A B (CDR LU))))))
)
(EARLYAB 'C 'F '(A B C D E F)) ; обращение к функции
T                                ; результат
```

Определим функцию под именем **ADDNL**, которая прибавляет к числовым элементам списка **L** число **N**.

```
(DEFUN ADDNL (N L)
  (COND ((NULL L) NIL)
```

```

      (T (CONS (+ N (CAR L)) (ADDNL N (CDR L)))))
)
(ADDNL 5 '(6 7 10))      ; обращение к функции
(11 12 15)                ; результат

```

Определим функцию под именем **LINEZN**, которая изображает горизонтальную линию с заданным числом выбранного знака.

```

(DEFUN LINEZN (N ZNAK)
  (COND ((= N 0) (PRINC))
        (T (PRINC ZNAK) (LINEZN (- N 1) ZNAK))))
)
(LINEZN 16 "***")        ; обращение к функции
*****                   ; результат

```

Вместо звездочки в функции (LINEZN...) можно использовать и другие символы, например, + и т. д.

Определим функцию под именем **MNOJL**, которая преобразует список в множество, то есть в список, в который каждый элемент входит только один раз.

```

(DEFUN MNOJL ( L )
  (COND ((NULL L) NIL)
        ((MEMBER (CAR L) (CDR L)) (MNOJL (CDR L)))
        (T (CONS (CAR L) (MNOJL (CDR L)))))
)
(MNOJL '(A B A D))        ; обращение к функции
(B A D)                   ; результат

```

Определим функцию под именем **DELLAST**, которая удаляет последний элемент списка.

```

(DEFUN DELLAST ( L )
  (COND ((NULL L) NIL)
        ((NULL (CDR L)) NIL)
        (T (CONS (CAR L) (DELLAST (CDR L)))))
)
(DELLAST '(A B C D E))    ; обращение к функции
(A B C D)                 ; результат

```

Определим функцию под именем **ASSOCL**, которая строит ассоциативный список.

```

(DEFUN ASSOCL (L1 L2)
  (MAPCAR 'CONS L1 L2)
)
(ASSOCL '(1 2 3) '(ONE TWO THREE)) ; обращение к функции
((1 . ONE) (2 . TWO) (3 . THREE))   ; результат выполнения

```


Определим функцию под именем **READL**, которая из множества вводимых элементов в строке образует список элементов.

```
(DEFUN READL ()
  (PRINC (STRCAT "(" (READ-LINE) ")"))
)
(READL) ; обращение к функции
AB C 1 2 34 ; вводимые данные с клавиатуры
(AB C 1 2 34) ; результат работы функции
```

Определим функцию под именем **READS**, которая считывает строку в виде списка из файла. Запрос названия диска, каталога и имени файла должен проводиться в диалоговом режиме.

```
(DEFUN READS ()
  (SETQ DD (GETSTRING "\n Введите имя диска - "))
  CC (GETSTRING "\n Введите имя каталога - ")
  EE (GETSTRING "\n Введите имя файла - ")
  F1 (STRCAT DD ":\\" CC "\\" EE )
  FF (EVAL (LIST "OPEN F1 " "r"))
  SS (READ (STRCAT "(" (READ-LINE FF) ")"))
  (CLOSE FF) )
(READS) ; обращение к функции
```

Допустим, файл под именем SETDGR.REZ содержится на диске C: в каталоге ORGANIZA. Первая строка файла выглядит так: 10 11. Тогда при обращении к функции под именем **READS** появится следующая запрашивающая информация. Справа приведены ответы:

Введите имя диска - C	; вводим имя диска C
Введите имя каталога - ORGANIZA	; вводим имя каталога
Введите имя файла - SETDGR.REZ	; вводим имя файла
(10 11)	; результат работы функции

Далее обратимся к функциям для выполнения различных вычислительных операций.

Определим функцию под именем **FACT** для расчета факториала с использованием рекуррентной формулы

$$N! = N * (N-1) * (N-2) * \dots * 3 * 2 * 1$$

или

$$N! = N * (N-1)!$$

```
(DEFUN FACT (N)
  (IF (= N 0) 1 (* N (FACT (- N 1)))))
)
(FACT 4) ; обращение к функции
24 ; результат
```

Одной из наиболее часто встречающихся операций при работе со списками является их соединение.

Определим функцию под именем **APPEND**, которая по двум спискам строит новый список, содержащий в определенной последовательности все элементы первого и второго списков.

```
(DEFUN APPEND (L1 L2)           ; соединение двух списков в один
  (IF (NULL L1) L2 (CONS (CAR L1) (APPEND (CDR L1) L2)))
)
(APPEND ' (A B C) ' (D E F G H)) ; вызов функции
(A B C D E F G H)                ; результат работы функции
```

Определим функцию под именем **MNOGST**, которая вычисляет степенной многочлен

$$Y = A_0 + A_1 X + A_2 X^2 + A_3 X^3 + \dots$$

Для вычисления степенного многочлена используем схему Горнера

$$Y = A_0 + X (A_1 + X (A_2 + X (A_3 + \dots)))$$

которая обеспечивает получение результата при минимальном числе операций.

```
(DEFUN MNOGST (X LA)           ; функция вычисления многочлена
  (SETQ Y 0
    N (LENGTH LA)              ; определение числа членов в многочлене
    I 0)
  (REPEAT (- N 1)
    (SETQ I (+ I 1)
      Y (* X (+ (NTH (- N I) LA) Y))))
  (SETQ Y (+ (NTH 0 LA) Y))
)
```

Запишем данную функцию в файл под именем MNOGST.LSP. Допустим, этот файл находится на диске C: в подкаталоге LISP каталога AUTOCAD. Обращение к этому файлу будет выглядеть так:

```
C:\\AUTOCAD\\LISP\\MNOGST.LSP
```

Далее в этом же файле запишем оператор для вывода результатов расчета в этот же файл.

```
(SETQ FW (OPEN "C:\\AUTOCAD\\LISP\\MNOGST.LSP" "a"))
```

Переменную FW часто называют дескриптором файла.

Вызовем в том же файле функцию под именем **MNOGST** для вычисления заданного степенного многочлена, например, многочлена вида:

$$Y = 0.5 + X + 2X^2 + 3X^3 + 4X^4 + 5X^5$$

для $X = 2$.

(MNOGST 2 '(0.5 1 2 3 4 5)) ; Обращение к функции

Запишем результат расчета в файл MNOGST.LSP.

(PRINT Y FW) ; вывод результатов в файл с дескриптором FW

Ниже представлен результат расчета, записанный в файл MNOGST.LSP.

258.5 ; результат

Определим функцию под именем **CHEB**, которая вычисляет полином Чебышева. Для вычисления полиномов Чебышева первого рода известна явная и рекуррентная формулы:

$$T_N(X) = N/2 \sum_{M=0}^{N/2} \frac{(-1)^M (N-M-1)!}{M!(N-2M)} (2X)^{N-2M}$$

$$T_{N+1}(X) = 2X T_N(X) - T_{N-1}(X)$$

В явной формуле верхний индекс суммирования означает ближайшее целое число, не превосходящее $N/2$.

Рекуррентную формулу чаще всего применяют при $N \geq 1$, полагая $T_0(X) = 1$ и $T_1(X) = X$.

(DEFUN CHEB (X N) ; Функция вычисления полинома Чебышева

(COND ((EQ N 0) 1)

((EQ N 1) X)

(T (- (* 2 X (CHEB X (- N 1))) (CHEB X (- N 2))))

)

)

Запишем данную функцию в файл под именем CHEB.LSP. Далее в этом же файле запишем оператор для вывода результатов расчета в этот же файл.

(SETQ FW (OPEN "C:\\ALISP\\CHEB.LSP" "a"))

Вызовем функцию под именем **CHEB** для вычисления полинома Чебышева с заданными параметрами и проведем вывод результатов расчета в файл с дескриптором FW.

(PRINT (CHEB 2 5) FW) ; обращение к функции

Ниже представлен результат расчета, записанный в файл CHEB.LSP.

362 ; результат

В отличие от языков высокого уровня, LISP, в том числе AutoLISP, предоставляет возможность конструирования функций языка самой программой и последующего ее выполнения. В качестве примера составим программу, которая сформирует функцию под именем **MAXD** для определения максимального из чисел, вводимых в диалоговом режиме.

```

(SETQ A (LIST MAX)) ; определение начального списка A
(SETQ N (GETINT "\n Введите число чисел в массиве. N = "))
I 1)
(REPEAT N
  (SETQ B (GETREAL (STRCAT "\n Введите " (ITOA I) "-е число: ")))
  I (+ I 1)
  A (CONS B A) ; формирование списка A
)
)
(SETQ MAXD (REVERSE A)) ; формирование функции в виде списка
(EVAL MAXD) ; выполнение функции

```

После загрузки файла в командной строке появятся запросы. Необходимо ввести нужную информацию, например:

```

Введите число чисел в массиве. N = 6
Введите 1-е число: 5.5
Введите 2-е число: 7
Введите 3-е число: 10.45
Введите 4-е число: 3.1
Введите 5-е число: 9
Введите 6-е число: 1.17
10.45 ; результат

```

Определим функцию под именем **MODES**, которая сохраняет значения системных переменных:

```

(DEFUN MODES (L)
  (SETQ MLST NIL)
  (REPEAT (LENGTH L)
    (SETQ MLST (APPEND MLST (LIST (LIST (CAR L)
      (GETVAR (CAR L))))))
    (SETQ L (CDR L)))
  )

```

Затем вызовем функцию **MODES** для сохранения определенных системных переменных, например:

```

(MODES '( "BLIPMODE" "GRIDMODE" "ORTHOMODE" "OSMODE"
  "SURFTAB1" "SURFTAB2" "UCSFOLLOW"))

```

Определим функцию под именем **MODER**, которая восстанавливает значения системных переменных:

```

(DEFUN MODER ()
  (REPEAT (LENGTH MLST)
    (SETVAR (CAAR MLST) (CADAR MLST))
    (SETQ MLST (CDR MLST)))
  )

```


2.5. Программирование на AutoLISP в среде Visual LISP

Visual LISP имеет ряд инструментальных средств и возможностей для облегчения разработки программ на языке AutoLISP.

При использовании Visual LISP разработка программы на языке AutoLISP включает следующие основные этапы:

- разработку последовательности действий решения задачи;
- разработку программы;
- форматирование программы для большей наглядности;
- поиск синтаксических ошибок в программе;
- тестирование и проверку работоспособности программы.

Ввод текста программы производится в окне консоли после подсказки `_S`. Развернутое окно консоли с введенной программой представлено на рис. 2.2.

Следует помнить, что текст, который вводится в окне консоли Visual LISP, а также любой результат выполнения можно сохранять для дальнейшего

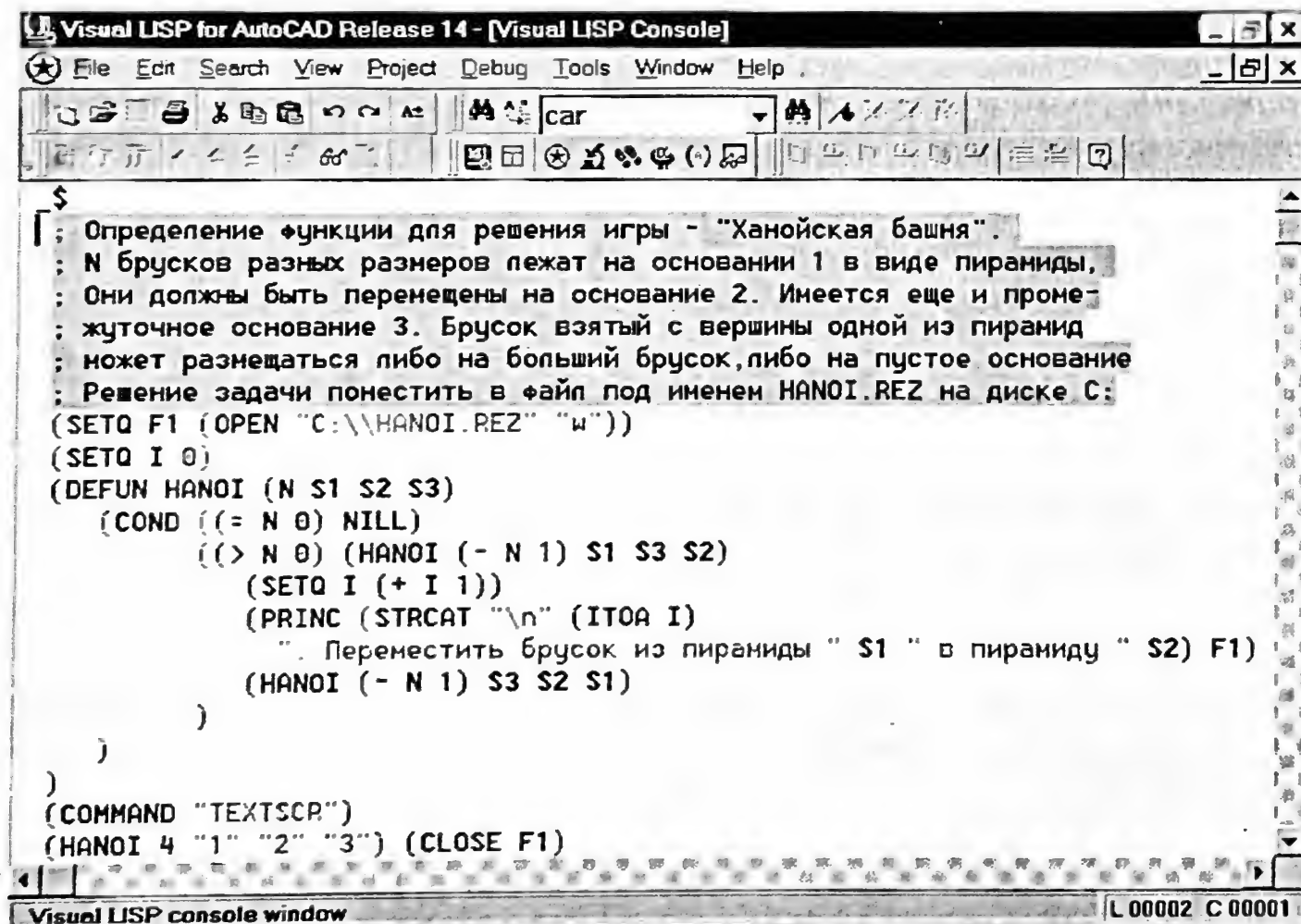


Рис. 2.2. Развернутое окно консоли Visual LISP

просмотра. Кроме того, любой текст в окне можно копировать и вставлять его в ответ на подсказку или в другое приложение Windows.

Консоль системы Visual LISP во многом схожа с командным окном AutoCAD, однако располагает рядом дополнительных возможностей. Например, чтобы отобразить текущее значение переменной AutoLISP в Visual LISP, достаточно напечатать имя переменной в окне консоли и нажать клавишу **Enter**, а чтобы просмотреть значение переменной в AutoCAD, необходимо напечатать имя переменной с восклицательным знаком впереди (!) и также нажать клавишу **Enter**.

Консоль AutoLISP имеет и другие дополнительные возможности:

- ввод выражения AutoLISP в строке, которая определяется нажатием комбинации клавиш **Ctrl+Enter**;
- ввод больше чем одного выражения перед нажатием клавиши **Enter**, и Visual LISP оценит каждое выражение перед выдачей результата на консоли.

Если курсор находится в окне консоли, но не на подсказке, то после нажатия клавиши **Enter** курсор перемещается в подсказку.

Если вы наберете текст в окне консоли прежде, чем нажмете **Enter** (например, результат предыдущей команды или предварительно введенного выражения), Visual LISP скопирует выбранный текст после подсказки.

Чтобы вернуться к предпоследнему введенному тексту в окне консоли, необходимо нажать клавишу **Tab**. Каждый раз после нажатия клавиши **Tab** ранее введенный текст заменяет текст в подсказке окна консоли. Как только процесс дойдет до первой введенной строки, он повторяется снова.

Комбинация клавиш **Shift+Tab** действует подобно **Tab**, но листает входную хронологию в обратном направлении.

Клавиша **Tab** допускает ассоциативный поиск во входной хронологии. Например, если вы напечатаете знак (+ в подсказке консоли и нажмете клавишу **Tab**, Visual LISP произведет поиск последнего текста, который начинался с символов (+, а если такого текста нет, то, возможно, появится звуковой сигнал. Комбинацию клавиш **Shift+Tab** можно использовать для того, чтобы осуществить ассоциативный поиск от ранних до более поздних вводов.

Клавиша **Esc** очищает окно консоли после подсказки. Нажатие комбинации клавиш **Shift+Esc** оставляет ранее введенный текст, но приводит к новой подсказке без выполнения ранее введенного текста.

Наиболее важные функции, необходимые при работе в окне консоли, сосредоточены в контекстном меню. Для быстрого вызова контекстного меню щелкните правой кнопкой мышки в любом месте окна консоли или нажмите комбинацию клавиш **Shift+F10** (рис. 2.3).

В зависимости от позиции курсора, а также от того, имеется ли выделенный текст в окне консоли, некоторые операции могут быть неактивными. Данное контекстное меню содержит следующие команды:

- **Cut** (Вырезать) – удаляет выделенный текст из окна и перемещает его в буфер;
- **Copy** (Копировать) – копирует выбранный текст в буфер обмена Windows (ClipBoard);
- **Paste** (Вставить) – вставляет содержимое буфера обмена в место расположения курсора;
- **Clear Console Window** (Очистить окно консоли) – очищает окно консоли;
- **Find** (Найти) – открывает диалоговое окно поиска задаваемого фрагмента;
- **Inspect** (Проверить) – открывает диалоговое окно просмотра выражения;
- **Add Watch** (Добавить наблюдение) – открывает диалоговое окно наблюдения выражения;
- **Apropos Window** (Окно поиска по фрагменту) – открывает диалоговое окно для настройки системы поиска слова по фрагменту;
- **Symbol Service** (Обслуживание символов) – открывает диалоговое окно обслуживания символов;
- **Undo** (Отменить) – отменяет последнюю выполненную команду;
- **Redo** (Восстановить) – восстанавливает последнюю отмененную команду;
- **AutoCAD Mode** (Режим AutoCAD) – передает весь ввод командной строке AutoCAD;
- **Toggle Console Log** (Установить/удалить файл регистрации) – открывает диалоговое окно **Open Log** (Открыть файл регистрации).

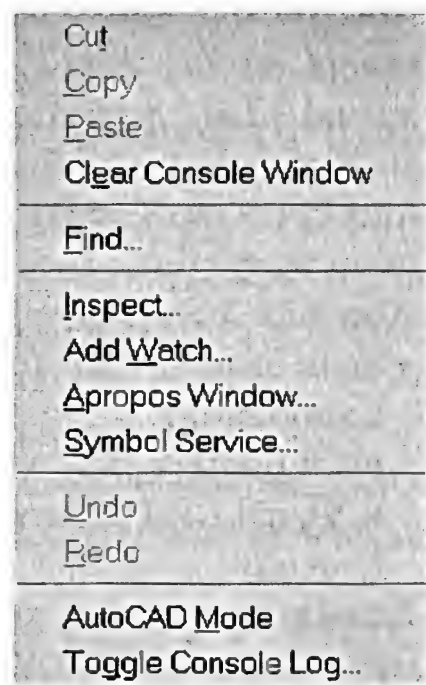


Рис. 2.3. Контекстное меню окна консоли

Обратите внимание, что между окном консоли Visual LISP и окном командных строк AutoCAD текст можно «вырезать и вставлять». Кроме того, окно консоли Visual LISP и окно командных строк AutoCAD отличаются по способу, которым обрабатываются клавиши **Space** и **Tab**. В окне консоли Visual LISP пробел не играет никакой роли и служит только разделителем. Если в окне командных строк AutoCAD нажать клавишу **Space** не внутри выражения, это вызовет окончание ввода и обработку текста, как если бы вы нажали клавишу **Enter** (Ввод).

Как только вы начнете вводить текст после подсказки в окне консоли, Visual LISP определяет, чем является введенное слово – встроенной

функцией AutoLISP, числом, строкой или другим элементом. Каждый тип элемента языка наделяется собственным цветом.

Если вы вводите имя функции в окне консоли после подсказки, то, щелкнув по кнопке **Help** (Помощь) на инструментальной панели **Tools** (Инструменты), получите справку Visual LISP о функции. Это правило действует в отношении любых элементов AutoLISP, Visual LISP или функции (ACTIVE), распознанных Visual LISP.

Вы можете сохранить запись всех действий в окне консоли, если регистрируете эти действия в Log (регистрационном) файле. Позже у вас будет возможность просмотреть файл и проанализировать действия, которые выполнялись в окне консоли.

Чтобы создать файл регистрации, выберите команду **Toggle Console Log** (Установить/удалить файл регистрации) падающего меню **File** (Файл) главного меню Visual LISP. При этом окно консоли должно быть активным. На экране появится диалоговое окно **Open Log** (Открыть файл регистрации) (рис. 2.4).

Выберите каталог для файла регистрации, а затем определите название файла. Если такой файл существует, на экране появится запрос о добавлении данных к уже имеющемуся файлу (рис. 2.5).

Допустим, файл регистрации уже существует (Log file C:\Мои документы\dom.LOG already exists). Если на вопрос **Append log to existing file?** (Присоединить регистрационные данные к существующему файлу?) вы ответите **Yes** (Да), Visual LISP добавит информацию с консоли к текущему

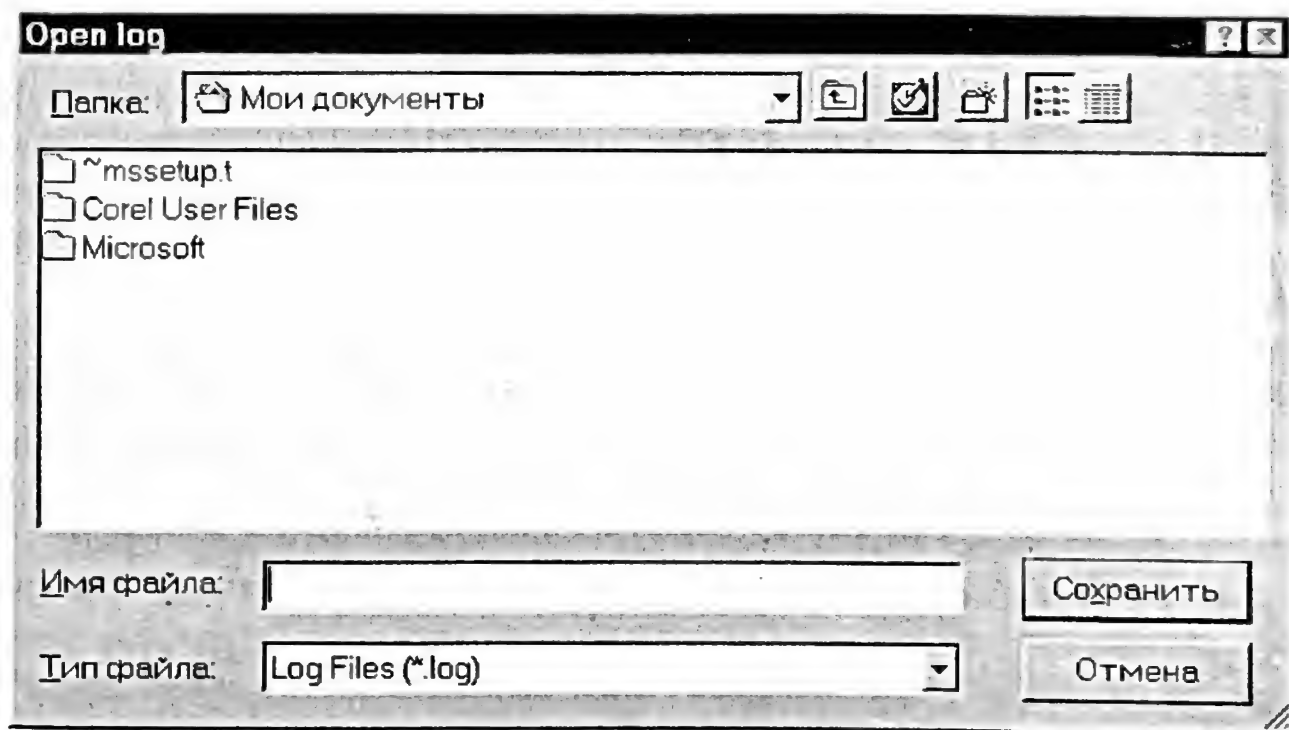


Рис. 2.4. Диалоговое окно открытия файла регистрации

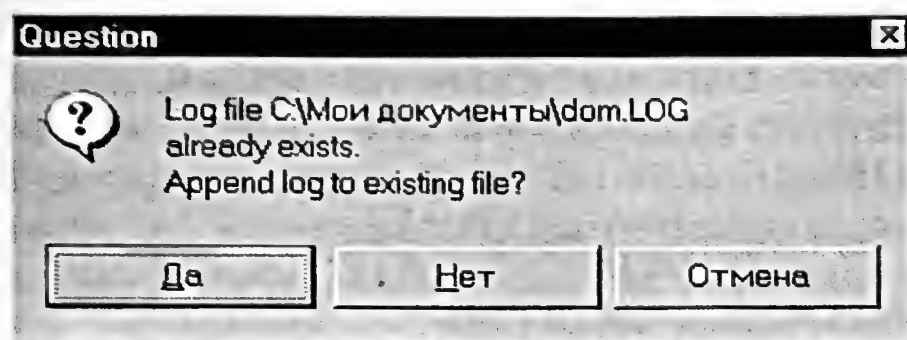


Рис. 2.5. Запрос о добавлении данных к существующему файлу регистрации

содержанию файла. В случае если вы предпочтете ответить **No** (Нет), Visual LISP также произведет запись в файл с этим же именем, однако первоначальное содержимое файла будет потеряно. Чтобы завершить операцию и присвоить файлу другое имя, нажмите кнопку **Отмена** (Cancel).

Для того чтобы завершить регистрацию и закрыть соответствующий файл, выберите пункт **Toggle Console Log** (Установить/удалить файл регистрации) из падающего меню **File** (Файл). По названию окна консоли можно определить, находится ли консоль в состоянии регистрации. Если регистрация проводится, то в названии окна отражено имя файла регистрации. Когда регистрация выключена, имя файла в названии окна консоли отсутствует.

Если вы забыли закрыть файл регистрации до выхода из Visual LISP, система сделает это автоматически. После закрытия файла регистрации вы можете просматривать его содержимое в любом текстовом редакторе типа Visual LISP.

Обратите внимание на возможность непосредственного взаимодействия через окно консоли AutoLISP с AutoCAD. Чтобы включить этот режим, выберите пункт **AutoCAD Mode** (Режим AutoCAD) из падающего меню **Tools** (Инструменты) главного меню Visual LISP или обратитесь к пункту **AutoCAD Mode** (Режим AutoCAD) в контекстном меню консоли. Подсказка в окне консоли изменится на **COMMAND**.

Любая последовательность символов, введенная после этой подсказки, будет рассматриваться как командная строка AutoCAD. Таким образом, у вас появится возможность вводить команды AutoCAD изнутри Visual LISP.

Рабочая среда AutoLISP отличается от рабочей среды Visual LISP.

Покажем это на следующих примерах.

```

_$ (SETQ X 1)           ; присваивает X значение 1 в среде Visual LISP
COMMAND: (SETQ Y 2)     ; присваивает Y значение 2 в среде AutoLISP
COMMAND: !X            ; отображает NIL в командной строке AutoCAD

```

```
_$ Y
NIL
_$ X
1
```

Чтобы вернуться к окну консоли Visual LISP, выберите пункт **AutoCAD Mode** (Режим AutoCAD) так же, как вы делали это при переключении в AutoCAD. Пункт меню AutoCAD Mode (Режим AutoCAD) работает как переключатель «вкл\выкл».

Не исключено, что возникнет ситуация, когда консоль Visual LISP будет ждать ввода из AutoCAD, а AutoCAD будет полагать, что управление передано Visual LISP. В таком случае можно попробовать прервать команду AutoCAD и вернуться в Visual LISP.

Если в окне консоли Visual LISP появится следующая подсказка:

```
1_$
; Entering keyboard break loop
```

необходимо возвратить управление AutoCAD и нажать клавишу **Esc** для перехода в рабочую среду Visual LISP. Чтобы возвратить управление AutoCAD, нажмите кнопку **Activate AutoCAD** (Активизировать AutoCAD) на инструментальной панели **View** (Просмотр).

После нажатия клавиши **Esc** в AutoCAD можно продолжать взаимодействие с консолью Visual LISP.

Текстовый редактор – один из основных компонентов Visual LISP. Он прост в использовании и располагает рядом возможностей, которые упрощают процесс программирования на AutoLISP. К таким возможностям текстового редактора относятся:

- выделение правильных выражений AutoLISP;
- отслеживание парных круглых скобок;
- выделение цветом различных элементов программы согласно синтаксису языка;
- выполнение правильных конструкций выражения AutoLISP без выхода из окна редактора.

Большинство команд текстового редактора вызываются из строки меню: для некоторых из них, используемых наиболее часто, предназначены кнопки инструментальной панели.

Чтобы открыть новый файл в текстовом редакторе Visual LISP, выберите пункт **New File** (Новый файл) из падающего меню **File** (Файл). На экране появится развернутое окно текстового редактора (рис. 2.6).

Чтобы ввести текст, наберите его в активном окне текстового редактора. Для начала новой строки нажмите клавишу **Enter**. Текстовый редактор не

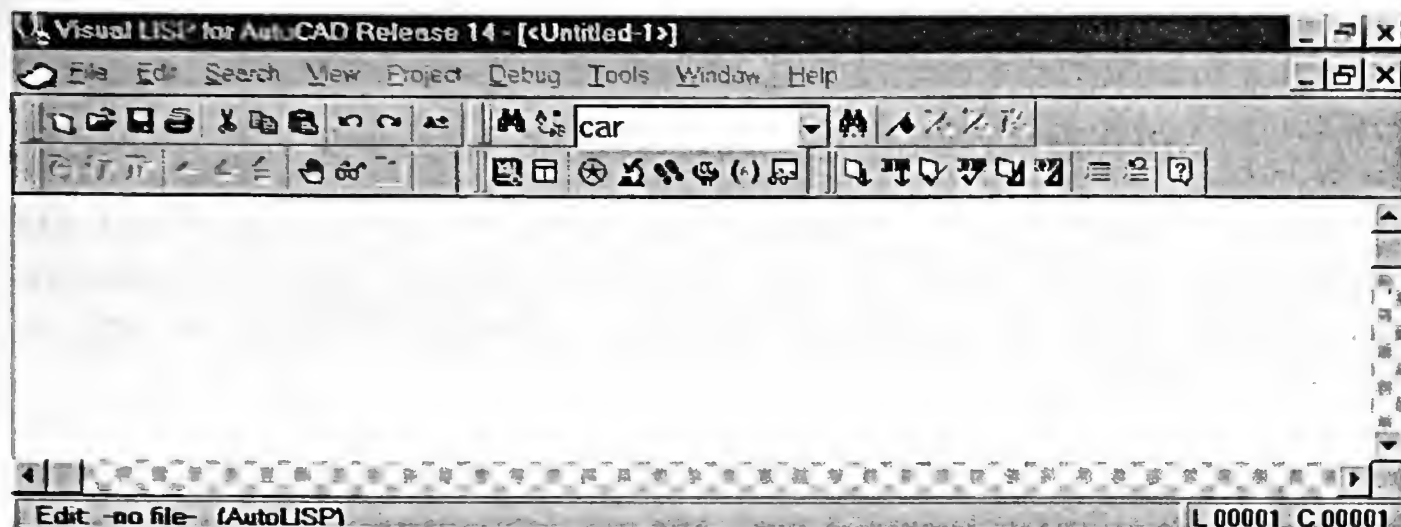


Рис. 2.6. Развернутое окно текстового редактора

переносит набранный текст на новую строку до тех пор, пока не будет нажата клавиша **Enter**.

Чтобы выровнять текстовую строку, введите пробелы или нажмите клавишу **Tab**. В последнем случае используйте **Code Formatter** (Кодовый форматор) Visual LISP, чтобы автоматически выравнивать выражения в программе.

Нажатие комбинации клавиш **Shift+Enter** очищает конечный пробел и символы табуляции, вставляет новую строку и отступы, использующие предыдущий непустой отступ строки.

Нажатие комбинации клавиш **Ctrl+Enter** очищает конечный пробел и символы табуляции, вставляет символ **NewLINE** и выполняет «интеллектуальный» отступ.

Чтобы вставить новый текст в уже существующий, щелкните мышкой в месте вставки и начните вводить текст. Место вставки можно указать также клавишами курсора.

Если вы хотите отменить только что введенный текст или последние изменения, выберите пункт **Undo** (Отменить) из падающего меню **Edit** (Редактирование) или щелкните мышкой по кнопке **Undo**. Команда **Undo** отменяет только последнее изменение.

Можно отменить также действие **Undo**. Для этого сразу после нажатия кнопки **Undo** (Отменить) выберите пункт **Redo** (Восстановить) из падающего меню **Edit** или щелкните мышкой по кнопке **Redo**.

Чтобы сохранить изменения в файле до следующего сеанса работы с Visual LISP, выберите пункт **Save As** (Сохранить как) из падающего меню **File** (Файл) главного меню Visual LISP или щелкните мышкой по кнопке **Save File** (Сохранить файл) и определите путь, а также имя файла. Если вы попытаетесь выйти из Visual LISP до сохранения файла, Visual LISP спросит, сохранить ли изменения.

Visual LISP поддерживает автоматическое создание резервных копий файлов, загруженных текстовым редактором. Фактически создание резервных копий происходит, когда вы сохраняете файл впервые. Файл с резервной копией имеет то же имя, что и исходный, за исключением расширения, которое включает символ подчеркивания _ и первые два символа первоначального расширения. Например, для исходного файла PRIVET.LSP имя файла с резервной копией выглядит следующим образом: PRIVET_1.S.

Для того чтобы файлы с резервными копиями создавались автоматически, необходимо выбрать пункт **General Options** (Общие параметры) из всплывающего меню пункта **Environment Options** (Параметры среды) падающего меню **Tools** (Инструменты). Режим автоматического создания резервной копии будет включен, если выбрать **Backup the file edited on first save** (Резервная копия файла, впервые отредактированная, сохраняется). Данный режим используется по умолчанию.

Чтобы открыть существующий файл, выберите пункт **Open File...** (Открыть файл) из падающего меню **File** (Файл) главного меню Visual LISP или щелкните по кнопке **Open File** (Открыть файл). На экране появится стандартное диалоговое окно **Open file to edit/view** (Открыть файл для редактирования/просмотра). Visual LISP открывает новое окно текстового редактора для файла, который вы выбираете. Открыть можно любое количество файлов, но работать можно только с одним – активным. Активное окно находится поверх остальных окон, и заголовок его отличается от них по цвету. Каждый файл Visual LISP помещает в свое отдельное окно текстового редактора. Имя выбранного файла находится в поле заголовка окна текстового редактора.

Если при выходе из Visual LISP файлы остаются открытыми, то они сохраняются до следующего сеанса работы в Visual LISP. Когда вы в следующий раз начинаете работу в Visual LISP, он автоматически открывает эти файлы.

По мере ввода программы в окне текстового редактора Visual LISP определяет, чем является введенное слово: встроенной функцией AutoLISP, числом, строковой константой или другим элементом языка, который он распознает. Каждому виду элемента присваивается собственный цвет, что помогает обнаружить отсутствующие кавычки или имя функции с орфографической ошибкой. Заданная по умолчанию схема цветов такова:

<i>Элементы языка AutoLISP</i>	<i>Цвет</i>
Встроенные функции	Голубой
Строки	Сиреневый
Целые числа	Зеленый

Вещественные числа

Комментарии

Круглые скобки

Переменные пользователя

Цвет морской волны

Сиреневый на сером фоне

Красный

Черный

Если функция неизвестна Visual LISP, она не будет идентифицирована и кодирована цветом.

Редактор Visual LISP обеспечивает кодирование цветом файлов LISP, а также DCL, SQL, C.

Visual LISP использует расширение имени файла, чтобы определить тип файла, и выбирает соответствующий цвет. Цвет, связанный с типом файла, можно изменять. Для этого необходимо выбрать пункт **Syntax Coloring...** (Синтаксис установки цвета) из всплывающего меню пункта **Window Attributes** (Параметры окна) из падающего меню **Tools** (Инструменты) главного меню Visual LISP.

Если вы выделяете имя функции в окне текстового редактора и нажимаете кнопку **Help** (Помощь) – кнопку со знаком вопроса на инструментальной панели **Tools** (Инструменты), появляется справка для данной функции Visual LISP.

Наиболее важные функции, необходимые при работе в окне текстового редактора, сосредоточены в контекстном меню. Для быстрого вызова контекстного меню щелкните правой кнопкой мышки в любом месте окна текстового редактора или нажмите комбинацию клавиш **Shift+F10** (рис. 2.7).

В зависимости от позиции курсора, а также оттого, имеется ли выделенный текст в окне текстового редактора, некоторые пункты могут быть неактивными (невыведенными) или даже отсутствовать.

Большинство пунктов контекстного меню окна текстового редактора аналогичны пунктам контекстного меню окна консоли (см. рис. 2.3), поэтому перечислим только новые из них:

Go to Last Edited (Идти к последнему месту редактирования) – переместить курсор в позицию последней точки редактирования;

Toggle Breakpoint (Установить/удалить точку прерывания в позиции курсора) – установить точку прерывания в позиции курсора, если ее там не было, и удалить ее, если она там была.

Если курсор стоит в позиции точки прерывания, то при вызове контекстного меню правой кнопкой мышки в нем вверху появляется новый пункт

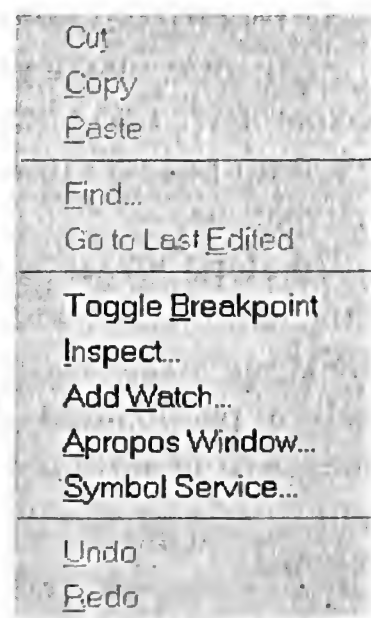


Рис. 2.7. Контекстное меню окна текстового редактора

Breakpoint service (Обслуживание точек прерывания). При выборе этого пункта контекстного меню окна текстового редактора появляется соответствующее диалоговое окно.

В редакторе текста Visual LISP термин «слово» означает последовательность символов, отделяемых одним или большим количеством следующих специальных символов:

Space	пробел
Tab	метка табуляции
'	одиночная кавычка
(левая круглая скобка
)	правая круглая скобка
"	удвоенная кавычка
;	точка с запятой
\n	перейти к новой строке (NewLine)
[]	непечатаемые символы ASCII (например, \001–\037)

Допустим, вы забыли правильное написание имени функции. Текстовый редактор Visual LISP может помочь восстановить слово двумя способами: по первым буквам (символам) слова или по фрагменту слова. Напечатайте, например, в окне текстового редактора два символа DE и нажмите комбинацию клавиш **Alt+ /** (**Alt+Slash**), чтобы восстановить полное слово по частично введенному. Получите имя **DEFUN**. Visual LISP восстанавливает частично введенное слово в соответствии со словом из таблицы идентификаторов Visual LISP. Чтобы вызвать список полных слов по частично введенному, нажмите комбинацию клавиш **Ctrl+Shift+ /**.

Таким образом, предоставляется возможность выбора щелчком мышки нужного слова.

Допустим, вы помните начальный фрагмент имени функции. Напечатайте, например, в окне текстового редактора **FUN**. Щелкните по кнопке **Apropos** (Поиск слова по фрагменту) – кнопке с буквой A в скобках. На экране появится окно **Apropos** (Поиск слова по фрагменту) (рис. 2.8).

В поле сообщений в нижней части окна имеется надпись **2 symbols starting with "fun"** (найжены 2 имени, начинающиеся с "fun").

Чтобы выбрать имя из окна, необходимо сделать следующее:

- выбрать из списка имя;
- щелкнуть правой кнопкой мышки для вывода контекстного меню (рис. 2.9);

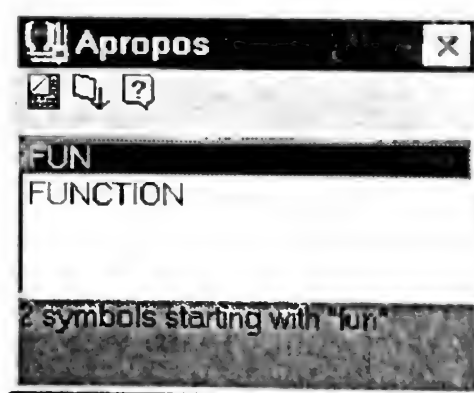


Рис. 2.8. Окно поиска списка слов по фрагменту **Apropos**

- выбрать пункт **Copy to clipboard** (Копировать в буфер обмена);
- щелкнуть в окне редактора в том месте, где необходимо вставить имя функции;
- щелкнуть правой кнопкой мышки для вывода контекстного меню (см. рис. 2.7), выбрать **Paste** (Вставить) или нажать комбинацию клавиш **Ctrl+V** для того, чтобы вставить текст.

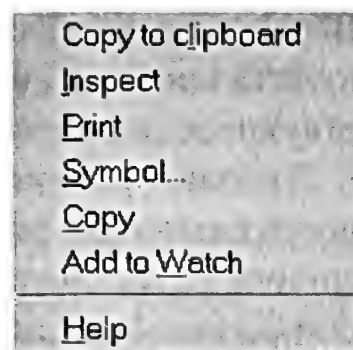


Рис. 2.9. Контекстное меню окна *Apropos*

Допустим, вы помните фрагмент имени функции. Напечатайте, например, в окне текстового редактора **DEFUN**. Щелкните по кнопке **Apropos** (Поиск слова по фрагменту) – кнопка с буквой **A** в скобках, на экране появится диалоговое окно **Apropos** (Поиск слова по фрагменту) (рис. 2.10).

Диалоговое окно поиска по фрагменту содержит следующие виды флажков:

- **Match by Prefix** – соответствовать префиксу;
- **Use WCMATCH** – использовать групповые символы, например, *;
- **Downcase Symbols** – соответствовать заданным символам.

Ниже слева имеются две кнопки:

- **Filter Value** (Фильтр данных) – открыть диалоговое окно выбора фильтра.
- **Filter Flags** (Фильтр флажков) – открыть диалоговое окно выбора флажков данных.
- Если щелкнуть по кнопке **Filter Value**, появится диалоговое окно выбора фильтра типа данных (рис. 2.11).

Диалоговое окно для выбора фильтра типа данных **Filter value type** (Фильтр типов данных) представляет собой перечень типов данных, среди которых может производиться поиск. Один из кружочков отмечается точкой. Для установки точки нужно щелкнуть по кружочку мышкой. В нашем примере поиск производится без фильтра, т.е. ищется полное слово среди всех типов данных.

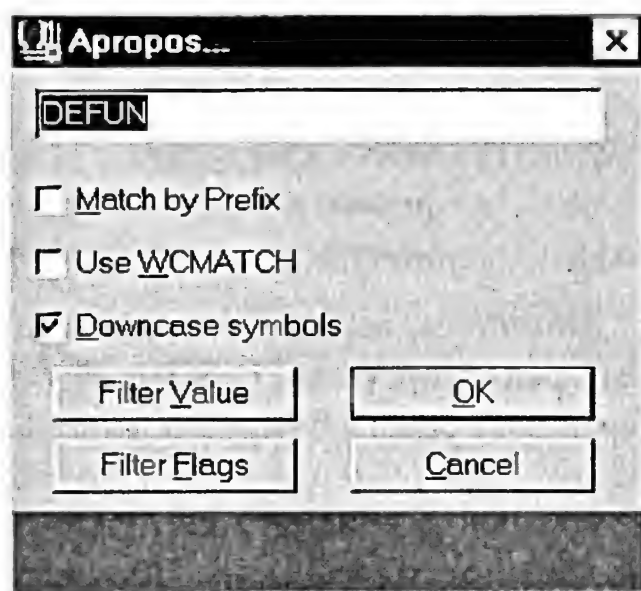


Рис. 2.10. Диалоговое окно поиска по фрагменту *Apropos*

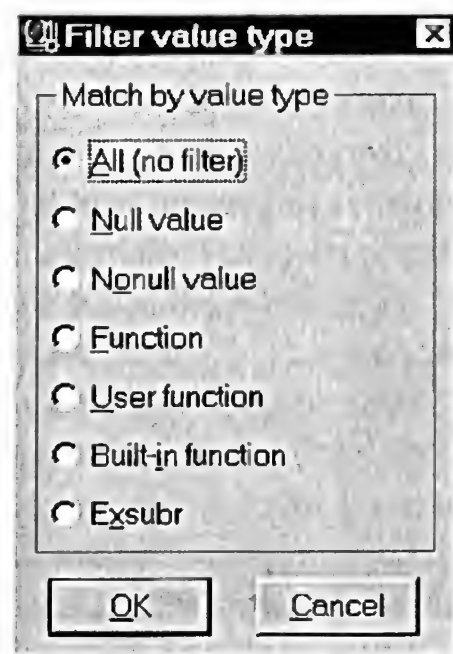


Рис. 2.11. Диалоговое окно для выбора фильтра типа данных

Поиск полного слова может производиться среди:

- **All (no filter)** – всех типов данных (без фильтра);
- **Null value** (Нулевые значения) – нулевых значений символов;
- **Nonull value** (Ненулевые значения) – ненулевых значений символов;
- **Function** (Функция) – функций (пользовательские, встроенные, ...);
- **User function** (Пользовательская функция) – определенных пользователем функций;
- **Built-in function** (Встроенная функция) – встроенных функций Visual LISP;
- **Exsubr** (Внешняя функция) – внешних функций.

Если щелкнуть по кнопке Filter Flags, появится диалоговое окно для выбора флажков (рис. 2.12).

Выбор имен производится в соответствии с установками флажков:

- **Protect Assign** – защитить присваивание;
- **Trace** – трассировка программы;
- **Debug on Entry** – отладка на входе;
- **Export to ACAD** – экспортирование в AutoCAD.



Рис. 2.12. Диалоговое окно для выбора флажков

Если фильтр флажка включен, рассматриваются только наборы символов с выбранными флажками.

После вызова пункта **Extra Commands** (Особые команды) падающего меню Edit или нажатия комбинации клавиш **Ctrl+E** при активном окне текстового редактора Visual LISP появится всплывающее меню особых команд редактирования (рис. 2.13), куда входят следующие команды:

- **Indent Block** (Сделать отступ для выделенного блока) – сделать отступ для каждой строки выделенного блока и добавить табуляцию к началу каждой строки;
- **Unindent Block** (Отменить отступ для выделенного блока) – отменить отступ для каждой строки выделенного блока текста и удалить табуляцию;
- **Prefix With** (Добавить в начале) – добавить текст, введенный в строке подсказки, в начало текущей строки или к каждой строке выделенного блока;

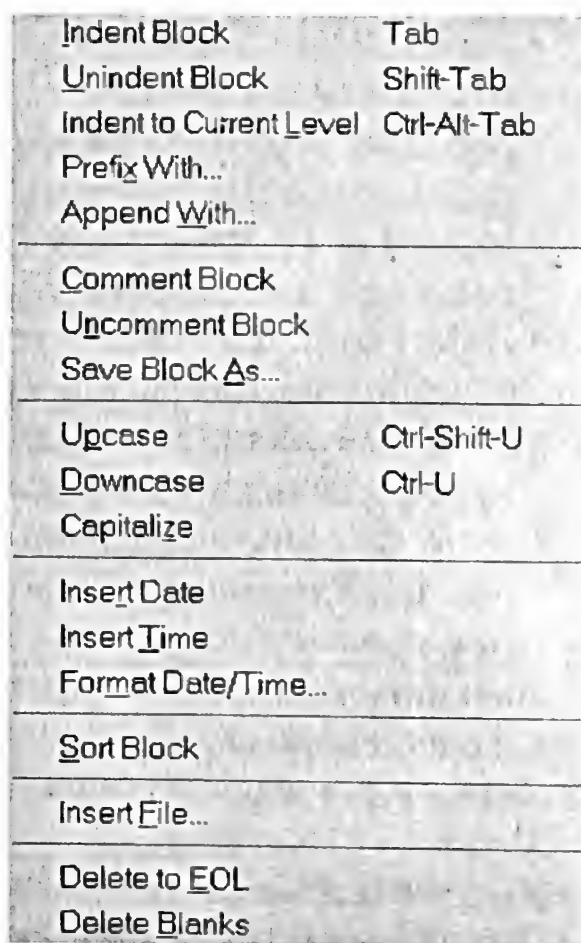


Рис. 2.13. Меню особых команд редактирования

- **Append With...** (Присоединить в конце) – присоединить текст, введенный в строке подсказки, в конце текущей строки или в конце каждой строки выделенного блока;
- **Comment Block** (Закомментировать выделенный блок) – преобразовать выделенный текст в комментарий;
- **Uncomment Block** (Разкомментировать выделенный блок) – преобразовать выделенные комментарии в текст программы;
- **Save Block As...** (Сохранить выделенный блок как файл) – сохранить выделенный блок текста как файл;
- **Uppercase** (Конвертировать в прописные) – преобразовать выделенный текст в прописные буквы;
- **Downcase** (Конвертировать в строчные) – преобразовать выделенный текст в строчные буквы;
- **Capitalize** (Представить слово с прописной буквы) – печатать прописными буквами первый символ каждого слова в выделенном тексте;
- **Insert Date** (Вставить дату) – вставить текущую дату (по умолчанию формат mm/dd/yy);
- **Insert Time** (Вставить время) – вставить текущее время (по умолчанию формат hh:mm:ss);
- **Format Date/Time** (Отформатировать дату/время) – изменить формат даты и времени;
- **Sort Block** (Отсортировать строки блока) – отсортировать строки в алфавитном порядке в выделенном блоке;
- **Insert File** (Вставить файл) – вставить текстовый файл в текущее окно редактора в позицию курсора;
- **Delete to EOL** (Удалить все от курсора до конца строки) – стереть все от позиции курсора до конца текущей линии;
- **Delete Blanks** (Удалить пробелы) – удалить все пробелы от позиции курсора до первого символа в строке.

Кроме клавиш управления курсором, для редактирования текста в Visual LISP можно использовать следующие комбинации клавиш:

<i>Перемещение</i>	<i>Клавиши и комбинации клавиш</i>
На одно слово влево	Ctrl+←
На одно слово вправо	Ctrl+→
К концу строки	End
К началу строки	Home
На одно окно вниз	Page Down
На одно окно вверх	Page Up
К концу документа	Ctrl+End
К началу документа	Ctrl+Home

К открывающей круглой скобке влево **Ctrl+[**
К закрывающей круглой скобке вправо **Ctrl+]**

Удалять слова, строки или абзацы можно, используя комбинации клавиш:

<i>Операции удаления</i>	<i>Комбинации клавиш</i>
Стирает слово слева от курсора	Ctrl+BackSpace
Стирает слово справа от курсора	Shift+BackSpace
Удаляет строку	Ctrl+Y

Использовать режим вставки можно путем нажатия клавиши **Ins**. В режиме вставки каждый напечатанный символ заменяет текущий.

К другим методам выделения текста относятся следующие:

<i>Операции выделения</i>	<i>Комбинации клавиш</i>
Расширение/сужение выделения:	
до следующей строки	Shift+↓
до предыдущей строки	Shift+↑
на одно окно вниз	Shift+Page Down
на одно окно вверх	Shift+Page Up
до следующего слова	Ctrl+Shift+→
до предыдущего слова	Ctrl+Shift+←
Расширение выделения до:	
конца строки	Shift+End
начала строки	Shift+Home
соответствующей круглой скобки слева	Ctrl+Shift+[
соответствующей круглой скобки справа	Ctrl+Shift+]
Перемещение курсора к другому концу выделения фрагмента	Alt+Enter

Кроме стандартных средств редактирования **Cut** (Вырезать), **Copy** (Скопировать) и **Paste** (Вставить), редактор Visual LISP располагает средствами перемещения текста внутри окна редактирования. Чтобы воспользоваться этими средствами, необходимо:

- подсветить (выделить) текст, который надо переместить;
- щелкнуть мышкой в любом месте внутри текста и, удерживая клавишу, переместить мышку, а с ней и текст в нужное место;
- отпустить клавишу мышки.

Для копирования текста необходимо перетащить его с помощью мышки при нажатой клавише **Ctrl**.

Выбранный текст можно скопировать в новый файл. Для этого необходимо нажать комбинацию клавиш **Ctrl+E**, выбрать пункт **Save Block As...** (Сохранить выделенный блок как). На экране появится диалоговое окно сохранения файла.

Visual LISP использует буфер обмена Windows для операций вырезания **Cut** (Вырезать) и копирования **Copy** (Копировать). Поэтому текст можно обменивать с любыми другими приложениями Windows, которые поддерживают эти функции. Кроме того, у пользователя есть возможность копировать и вставлять текст между окнами текстового редактора и окном консоли Visual LISP.

Структурную запись программы лучше всего проводить с помощью форматера Visual LISP.

Чтобы выровнять выбранные строки программы, нажмите клавишу **Tab** или комбинацию клавиш **Ctrl+E**, выберите размер отступа. Visual LISP вставляет символ табуляции в начале каждой строки, которую вы выбрали.

Величиной отступа символа табуляции можно управлять, если выбрать пункт **Configure Current** (Текущая конфигурация) всплывающего меню **Window Attributes** (Параметры окна) падающего меню **Tools** (Инструменты) и в появившемся диалоговом окне **Window attributes** (Параметры окна) (рис. 2.14.) установить значение **Tab Width** (Ширина отступа).

Редактор текста Visual LISP обеспечивает эффективный поиск элемента текста. Для того чтобы начать поиск, выберите пункт **Find** (Найти) из падающего меню **Search** (Поиск) или щелкните по кнопке **Find** – кнопке с изображением бинокля, либо нажмите комбинацию клавиш **Ctrl+F**. На экране появится диалоговое окно **Find** (Найти) (рис. 2.15).

В текстовом поле **Find What:** (Найти что) наберите с клавиатуры строку символов, которую предстоит найти.

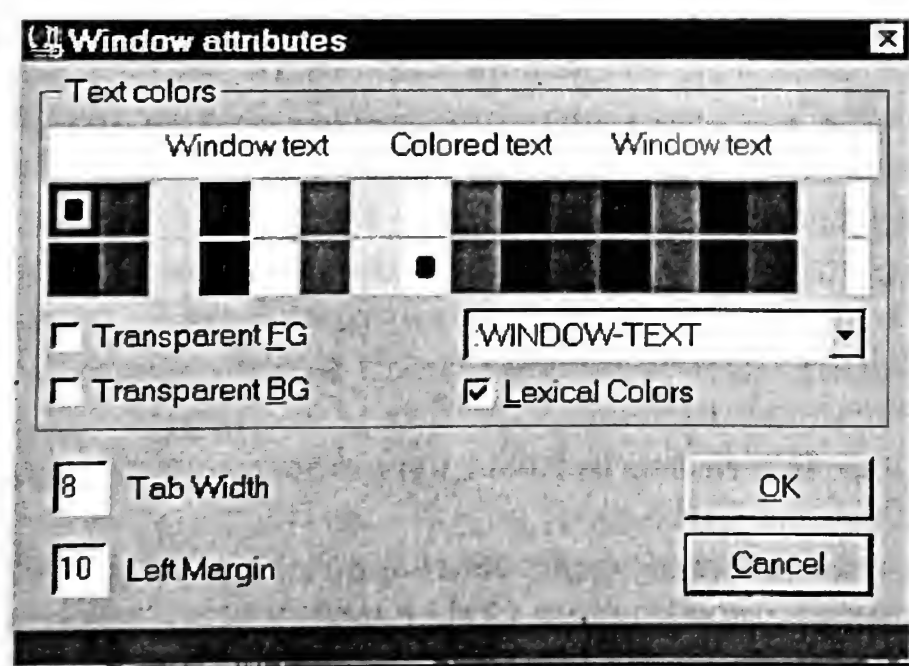


Рис. 2.14. Диалоговое окно параметров окна

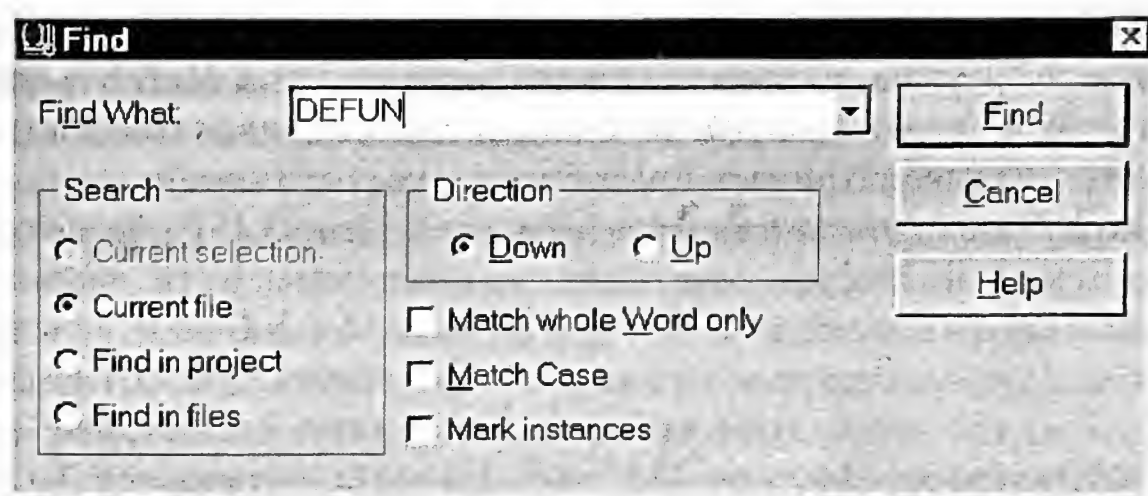


Рис. 2.15. Диалоговое окно поиска

Можно предварительно в окне текстового редактора Visual LISP выделить фрагмент текста, который предстоит найти, и после этого начать поиск. Тогда в диалоговом окне поиска Find (Найти) выделенный текст будет помещен в поле **Find What:** (Найти что) автоматически.

Для ограничения области поиска используется переключатель **Search** (Поиск). Он предоставляет несколько вариантов поиска, из которых можно выбрать только один:

- **Current selection** (Текущий выбор) – Visual LISP обеспечивает поиск только в выделенном тексте, который находится в окне текстового редактора;
- **Current file** (Текущий файл) – обеспечивает поиск только в текущем файле;
- **Find in project** (Найти в проекте) – Visual LISP запрашивает имя проекта, чтобы осуществлять поиск во всех его файлах. Результаты поиска отображаются в новом окне;
- **Find in files** (Найти в файлах) – Visual LISP позволяет определить каталог (папку) Window для поиска текста. Если задать поиск во всех подкаталогах указанного каталога, то Visual LISP просмотрит все файлы и представит результаты поиска.

Для ограничения направления поиска используется переключатель **Direction** (Направление). Он предоставляет два варианта поиска, из которых можно выбрать только один:

- **Down** – от позиции курсора вниз к концу файла;
- **Up** – от позиции курсора вверх к началу файла.

Параметры поиска могут быть заданы с помощью следующих флажков:

- **Match whole Word only** (Соответствовать только целому слову). Если флажок включен, Visual LISP обеспечивает поиск только слов, написанных полностью. Когда, например, осуществляется поиск слова

"ENT", то найденное в тексте слово "Enter" Visual LISP не рассматривает как результат поиска. Однако, если флажок **Match whole Word only** не включен, Visual LISP считает, что ENT является частью слова "Enter", и рассматривает Enter в качестве результата поиска;

- **Match Case** (Соответствовать регистру). Visual LISP обеспечивает поиск текста, полностью совпадающего с заданным. Если, например, осуществляется поиск слова "ENT" и флажок Match Case включен, то слово "ent" не рассматривается в качестве результата поиска. Когда флажок не включен, то результатом поиска может быть также слово "ent";
- **Mark instances** (Метить образцы). Закладка, установленная в тексте, будет добавлена к **Mark Ring** (Кольцо закладок), что позволит быстро вернуться к этой закладке в дальнейшем. Для начала поиска щелкните по кнопке **Find** (Найти). Чтобы продолжить поиск, используйте функциональную клавишу **F3**. Для прекращения поиска нажмите кнопку **Cancel** (Отменить).

Visual LISP сохраняет каждую строку поиска, которая введена в текстовом поле **Find What:** (Найти что) (рис. 2.16).

Чтобы повторить поиск строки, введенной в поле Find What ранее, достаточно выбрать необходимую строку в раскрывающемся списке Find What. Для этого щелкните по стрелке, подсветите нужную строку и нажмите кнопку **Find** (Найти).

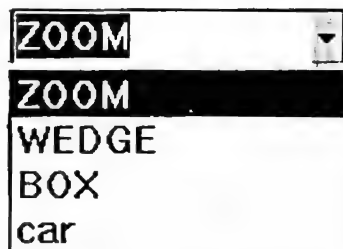


Рис. 2.16.

Раскрывающийся
список строк поиска

Падающее меню **Search** (Поиск) содержит пункт **Replace** (Заменить), который позволяет заменить текст поиска на определенную вами текстовую строку (рис. 2.17).

Диалоговое окно поиска и замены Find очень похоже на диалоговое окно поиска (см. рис. 2.15). Отличие между ними состоит в том, что окно

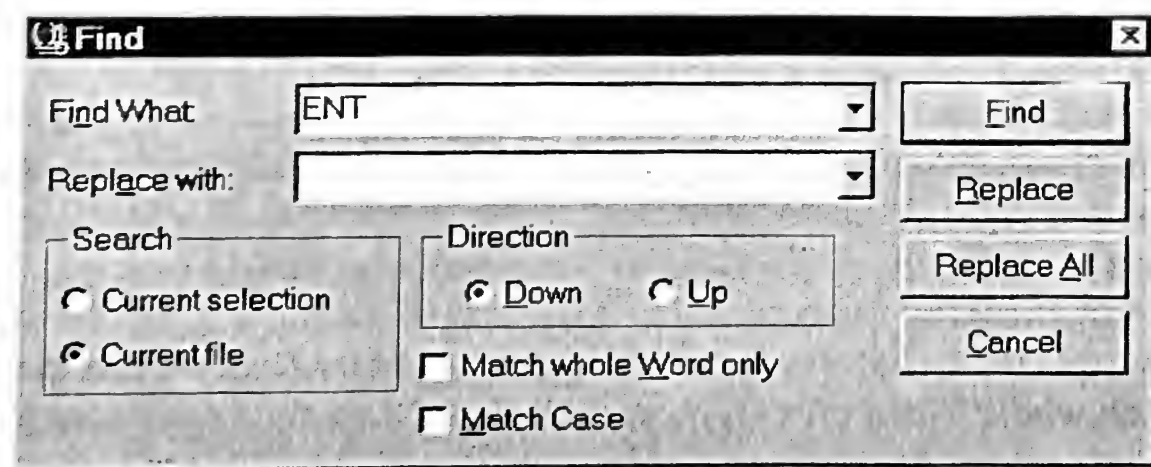


Рис. 2.17. Диалоговое окно поиска и замены Find

поиска и замены Find содержит дополнительное текстовое поле **Replace with:** (Заменить на). В этом поле вводится текст, на который Visual LISP должен заменить заданный текст. Для выполнения поиска из диалогового окна поиска и замены Find можно воспользоваться следующими кнопками:

- **Find** (Найти) – нахождение следующего местонахождения строки поиска;
- **Replace** (Заменить) – замена найденного текста на строку замены;
- **Replace All** (Заменить все) – замена всех найденных строк поиска на строку замены;
- **Cancel** (Отменить) – отмена замены. Если какую-либо из найденных строк поиска вы не хотите заменять, щелкните по кнопке **Find** (Найти), чтобы продолжить поиск следующего местонахождения текста, или **Cancel** (Отменить), чтобы закончить поиск.

Visual LISP имеет специальное средство для пошагового перемещения по тексту – набор закладок (Bookmarks). Это средство (до 32 закладок в каждом окне) помогает быстро перемещаться по тексту в редакторе, консоли и других текстовых окнах. Каждое окно поддерживает свой собственный набор закладок, а инструментальные средства передвижения по закладкам допускают переход к закладке внутри каждого окна независимо от других окон. Набор закладок внутри окна известен как кольцо (RING) закладок. Можно шагать по кольцу (набору) закладок вперед или назад и в конечном счете вернуться к начальной закладке.

Всякий раз при переходе от закладки к закладке Visual LISP временно удаляет закладку, к которой производится переход, и помещает в начало строки курсор. Чтобы установить/удалить закладку, переместите курсор в строку установки/удаления закладки. Щелкните по кнопке **Toggle bookmark** (Установить/удалить закладку) – кнопке с синим флагом на инструментальной панели **Search** (Поиск) или нажмите комбинацию клавиш **Alt+.** (точка).

Если использовать пункт **Find** (Найти) для поиска текста, закладки могут быть выставлены автоматически.

Когда текущее окно содержит закладки, вы можете:

- переместить курсор в строку, содержащую предыдущую закладку в кольцо. Для этого необходимо выбрать пункт **Previous Bookmarks** (Предыдущие закладки) из всплывающего меню **Bookmarks** (Закладки) падающего меню **Search** (Поиск) или нажать кнопку **Previous Bookmarks** (Предыдущие закладки) на инструментальной панели. Можно также нажать комбинацию клавиш **Ctrl+,** (запятая);
- переместить курсор в строку, содержащую следующую закладку в кольцо. Для этого выберите пункт **Next Bookmarks** (Следующие

закладки) из всплывающего меню **Bookmarks** (Закладки) падающего меню **Search** (Поиск) или нажмите кнопку **Next Bookmarks** (Следующие закладки) на инструментальной панели. Можно также нажать комбинацию клавиш **Ctrl+.** (точка).

Можно выбирать текст и между двумя закладками:

- чтобы выбрать текст между текущим расположением закладки и предыдущим, необходимо нажать комбинацию клавиш **Ctrl+Shift+.** (запятая);
- текст между текущим расположением закладки и следующим можно выбрать, если нажать комбинацию клавиш **Ctrl+Shift+.** (точка).

Для удаления одиночной закладки переместите курсор в строку, в которой расположена закладка. Щелкните по кнопке **Toggle Bookmark** (Установить/удалить закладку) на инструментальной панели **Search** (Поиск) или нажмите комбинацию клавиш **Alt+.** (точка). Пункт **Toggle Bookmark** работает как переключатель «вкл/выкл». Если вы выбираете пункт, когда закладка установлена, **Toggle Bookmark** выключает ее. Воспользуйтесь этой командой, когда нет закладок, и **Toggle Bookmark** (Установить/удалить закладку) вставит закладку.

Для удаления всех закладок в вашей программе нажмите на инструментальной панели кнопку **Clear All Bookmark** (Очистить все закладки) или выберите пункт **Clear All** (Очистить все) из всплывающего меню **Bookmarks** (Закладки) падающего меню **Search** (Поиск).

Visual LISP имеет специальное средство для структурирования (форматирования) программ – формater. Формater Visual LISP преобразует выражения AutoLISP в красивый удобочитаемый вид.

Для форматирования всего текста в активном окне редактора выберите пункт **Format AutoLISP in Editor** (Отформатировать программу в активном окне редактора) падающего меню **Tools** (Инструменты) главного меню Visual LISP или щелкните по кнопке **Format Edit Window** (Отформатировать программу в активном окне редактора) на инструментальной панели **Tools** (Инструменты).

Для форматирования части программы выделите фрагмент программы и выберите пункт **Format AutoLISP** падающего меню **Tools** (Инструменты) или щелкните по кнопке **Format Selection** (Форматирование выделенного фрагмента) инструментальной панели **Tools** (Инструменты). Если выделенный фрагмент выражений AutoLISP имеет ошибки, формater выдаст сообщение об ошибке.

Допустим, вы ввели в окне текстового редактора выражение AutoLISP.

```
(SETQ A 10 B 20 C 40
```

Но в этом выражении вы случайно забыли поставить закрывающую круглую скобку. Затем вы выделили это выражение и запустили процедуру форматирования, щелкнув по кнопке **Format Selection** (Форматирование выделенного фрагмента). Появится окно с сообщением и вопросом (рис. 2.18).

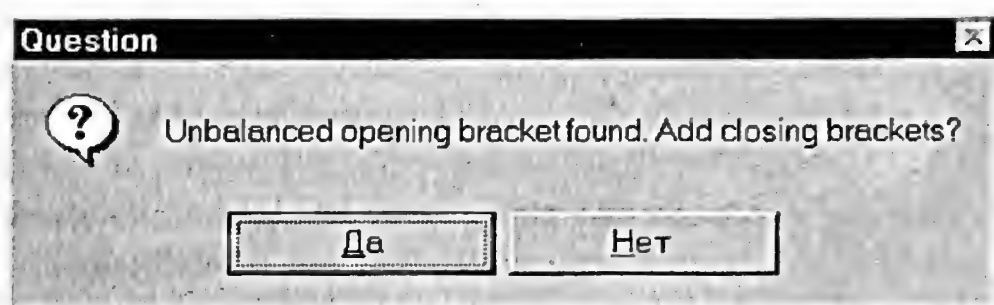


Рис. 2.18. Сообщение формatera

Допустим, в ответ на сообщение и вопрос **Unbalanced opening bracket found. Add closing brackets?** (Обнаружена несбалансированная открывающая круглая скобка. Добавить закрывающую круглую скобку?) вы щелкнули по кнопке **Yes** (Да), появится окно с таким подтверждением:

1 closing brackets were added while formatting (1 закрывающая круглая скобка была добавлена в процессе форматирования).

Следует щелкнуть по кнопке **ОК**. Появится результат форматирования введенного и выделенного выражения:

```
(SETQ  A 10  
       B 20  
       C 40)
```

Если хотите установить круглые скобки самостоятельно, нажмите кнопку **No** (Нет).

Форматер Visual LISP позволяет задать вид форматирования. Для этого необходимо вызвать пункт **AutoLISP Format Options...** (Параметры форматирования программ AutoLISP) из всплывающего меню, а затем пункт **Environment Options...** (Параметры среды) из падающего меню **Tools** (Инструменты). На экране появится диалоговое окно **AutoLISP Format options** (Параметры форматирования AutoLISP) (рис. 2.19).

Существуют два основных стиля форматирования программ:

- однострочный;
- многострочный – широкий, узкий или в столбец.

В соответствии с однострочным стилем все элементы программы размещаются в одной строке и разделяются одиночным пробелом.

Согласно многострочному широкому стилю первый элемент размещается в той же строке, что и имя функции, а остальные элементы выравниваются в столбце под первым элементом.

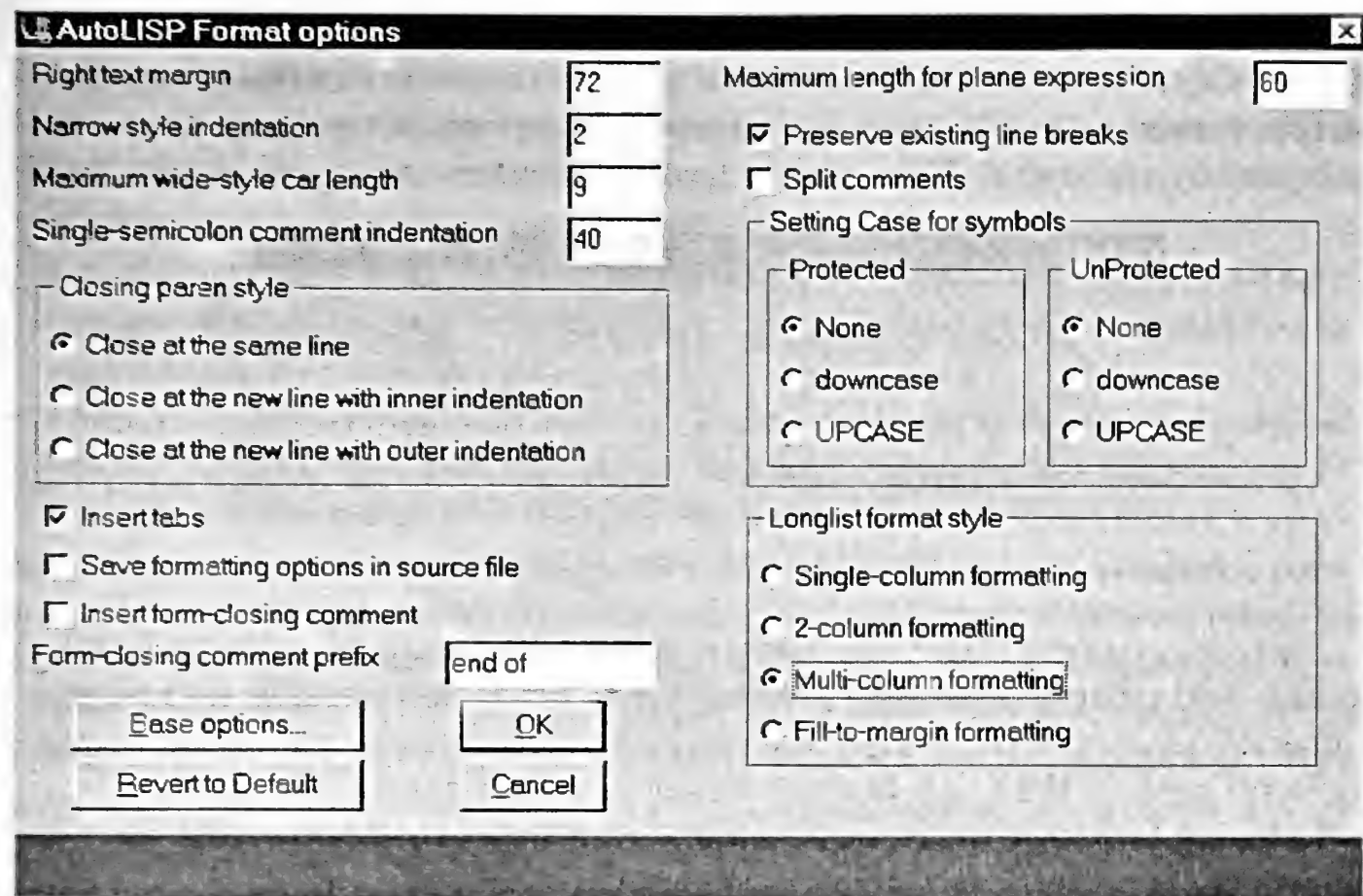


Рис. 2.19. Диалоговое окно выбора стиля форматирования

Многострочный узкий стиль предполагает размещение первого элемента в следующей строке после имени функции и выравнивание остальных элементов в столбце под первым элементом. Смещение позиции первого элемента относительно исходной позиции выражения управляется с помощью параметра **Narrow style indentation** (Суженный стиль выравнивания).

Многострочный стиль в столбец – это стиль, в котором все элементы размещены в столбец.

В дополнение к основным стилям форматирования можно устанавливать и другие параметры форматирования. Рассмотрим различные режимы управления позицией расположения круглой заключительной скобки для многострочных стилей форматирования, из которых может быть выбран только один:

- **Close at the same line** (Закрыть на той же самой строке) – закрытие круглой скобкой на последней строке каждого выражения;
- **Close at the new line with inner indentation** (Закрыть в новой строке с внутренним выравниванием) – закрытие круглой скобки на следующей строке после последней строки каждого выражения с внутренним выравниванием;
- **Close at the new line with outer indentation** (Закрыть в новой строке с внешним отступом) – закрытие круглой скобки на следующей (после

последней строки каждого выражения) строке с внешним выравниванием.

Текстовое поле **Form-closing comment prefix** (Завершающая форма комментария) позволяет добавить комментарий после выражения и используется только с последними двумя режимами. **Formatter** вставляет комментарий вида:

; _ END OF <имя функции>

Включенный флажок **Preserve existing line breaks** (Сохранить существующие разрывы строк) позволяет сохранить разрыв строк.

Флажок **Split comments** (Разбить комментарии) позволяет разбить длинный комментарий, который выходит за пределы правого поля.

Форматер Visual LISP распознает пять видов комментариев и позиционирует их следующим образом:

;| *Встроенный комментарий* |; – включается в любое место программы, задается внутри символов ; | ... |;

; *Комментарий после точки с запятой* – вставляется в конец текущей строки;

;; *Комментарий в текущем ряду* – пишется, начиная с новой строки с текущей позиции предыдущего выражения, строки выравниваются по последней строке программы.

;;; *Заголовок или комментарий без отступа.*

; _ *Функционально заключительный комментарий* – комментарий в конце последнего выражения.

Рассмотрим различные виды комментариев на следующем примере.
Исходный текст:

```
(DEFUN LIST123 (X)
```

```
;| встроенный комментарий |;
```

```
(LIST 1 2 3) ; комментарий после точки с запятой
```

```
;; комментарий в текущем ряду
```

```
;;; заголовок или комментарий без отступа
```

```
) ; _ функционально заключительный комментарий
```

Отформатированный текст:

```
( DEFUN LIST123 (X);| встроенный комментарий |;
```

```
(LIST 1 2 3) ; комментарий после точки с запятой
```

```
;; комментарий в текущем ряду
```

```
;;; заголовок или комментарий с позиции начального выражения
```

```
) ; _ функционально заключительный комментарий
```

Чтобы сохранить установки параметров форматтера Visual LISP, выберите пункт **Save Settings** (Сохранить установки) из падающего меню **Tools** (Инструменты) интегрированной среды Visual LISP. Текущие установки можно сохранить также в активном текстовом окне редактора Visual LISP.

Выберите пункт **Environment options** (Параметры среды) падающего меню **Tools** (Инструменты), а затем пункт **AutoLISP Format options** (Параметры форматирования AutoLISP) из всплывающего меню.

Размер окна и табуляция соответствуют установкам Windows. Установки можно изменить, если выбрать пункт **Window Attributes** (Атрибуты окна) падающего меню **Tools**, а затем пункт **Configure Current** (Текущая конфигурация) всплывающего меню. Форматер доступен только из окна редактора Visual LISP.

Одним из основных преимуществ Visual LISP является наличие различных средств отладки. Данные средства позволяют следить за тем, что происходит во время выполнения программы. При этом можно выбрать любой ее фрагмент. Не менее важно и то, что Visual LISP обеспечивает ряд возможностей для обнаружения ошибок прежде, чем программа начнет выполняться.

По сравнению с другими языками программирования AutoLISP использует круглые скобки наиболее часто. Как уже отмечалось, одна из самых распространенных синтаксических ошибок в AutoLISP – несоответствие количества открытых круглых скобок количеству закрытых. Visual LISP располагает рядом инструментальных средств, необходимых для обнаружения несоответствия между круглыми скобками.

Форматер Visual LISP осуществляет поиск незакрытых скобок во время форматирования программы. Он может добавить круглые скобки там, где считает нужным, – как правило, в конце программы.

Чтобы определить, правильно ли установлены круглые скобки, необходимо проверить структуру вашей программы. Для поиска парных круглых скобок можно использовать один из пунктов всплывающего меню пункта **Parentheses Matching** (Соответствие круглых скобок) падающего меню **Edit** (Редактирование):

- **Match Forward (Ctrl+J)** (Переместить курсор в конец выражения) – перемещает курсор в позицию за соответствующей закрывающей круглой скобкой, которая соответствует открывающей круглой скобке. Если близлежащего соответствия нет, курсор перемещается на один шаг вверх в иерархии вложения функций.
- **Match Backward (Ctrl+[)** (Переместить курсор в начало выражения) – перемещает курсор в позицию соответствующей открывающей круглой скобки. Если такой нет, курсор перемещается на один шаг вверх в иерархии вложения функций.
- **Select Forward (Ctrl+Shift+J)** (Переместить курсор в конец выражения и выделить его) – перемещает курсор так же, как и команда Match

Forward, а кроме того, выделяет весь текст между начальной и конечной позициями.

- **Select Backward (Ctrl+Shift+[)** (Переместить курсор в начало выражения и выделить его) – перемещает курсор так же, как и команда Match Backward, и, кроме того, выделяет весь текст между начальной и конечной позициями.

Если выбрать пункт **Check text in Editor** (Проверить синтаксис текста) падающего меню **Tools** (Инструменты) главного меню Visual LISP, можно провести дополнительную проверку синтаксиса программы.

Для проверки синтаксиса фрагмента программы используйте пункт **Check Selection** (Проверка синтаксиса выделенного текста) из падающего меню **Tools**.

Чтобы выполнить программу из окна текстового редактора, выберите пункт **Load text in editor** (Загрузить текст из редактора) падающего меню **Tools** главного меню Visual LISP или нажмите кнопку **Load active edit window** (Загрузить текст из активного окна редактора) на инструментальной панели **Tools**. На экране появится сообщение о загрузке программы на консоль.

Чтобы выполнить программу, введите имя функции в круглых скобках. Например:

```
_1$ (PRIVET)
```

Вы можете выполнить только часть программы. Для этого необходимо выделить нужную часть программы, выбрать пункт **Load Selection** (Загрузить выделенный текст) падающего меню **Tools** (Инструменты).

Рассмотрим процесс загрузки файлов с расширением LSP в системе AutoCAD. Создайте файл, например, PRIVETW.LSP и наберите в нем следующую программу:

```
(DEFUN PRIVET (IMY)
  (COMMAND
    "STYLE" "" "TIMES NEW ROMAN CYR" "60" "1" "" "" "" ""
    "TEXT" "20,1500" "0" (STRCAT "ДОРОГОЙ " IMY " !!!")
    "TEXT" "3,100" "17" " АВТОР ВАС ПРИВЕТСТВУЕТ "
    "TEXT" "250,50" "0" " УСПЕХА ВАМ !!!"
    "ZOOM" "A") ; Текст приветствия будет размещен во весь экран
  )
(PRIVET "Пользователь")
```

Командная строка AutoCAD для загрузки файла будет выглядеть следующим образом:

```
COMMAND: (LOAD "C:\PRIVETW")
```

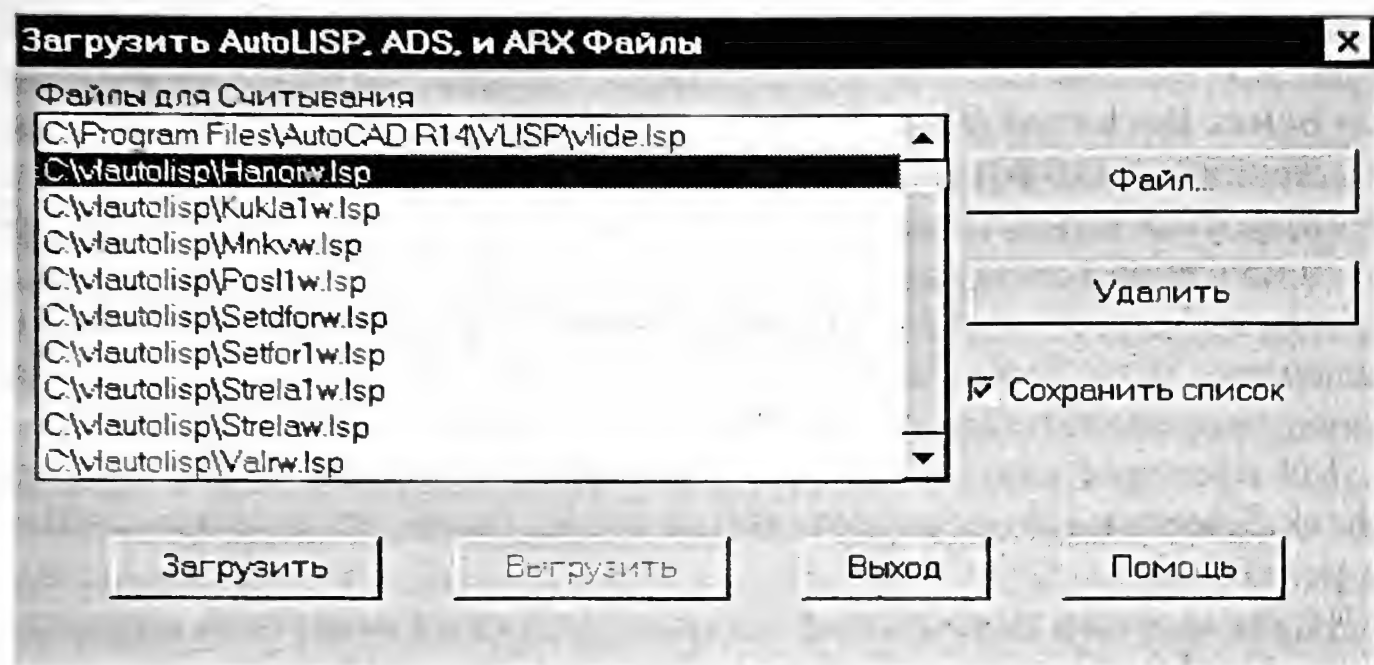



Рис. 2.20. Диалоговое окно загрузки файлов

Ждите ответ на экране монитора. Успеха Вам!!!

Файл PRIVETW.LSP можно ввести также, если использовать диалоговое окно **Загрузить AutoLISP, ADS, и ARX Файлы** (Load AutoLISP, ADS, and ARX Files) (рис. 2.20). Для этого выберите пункт **Load Application...** (Загрузить приложение) в падающем меню **Tools** (Инструменты).

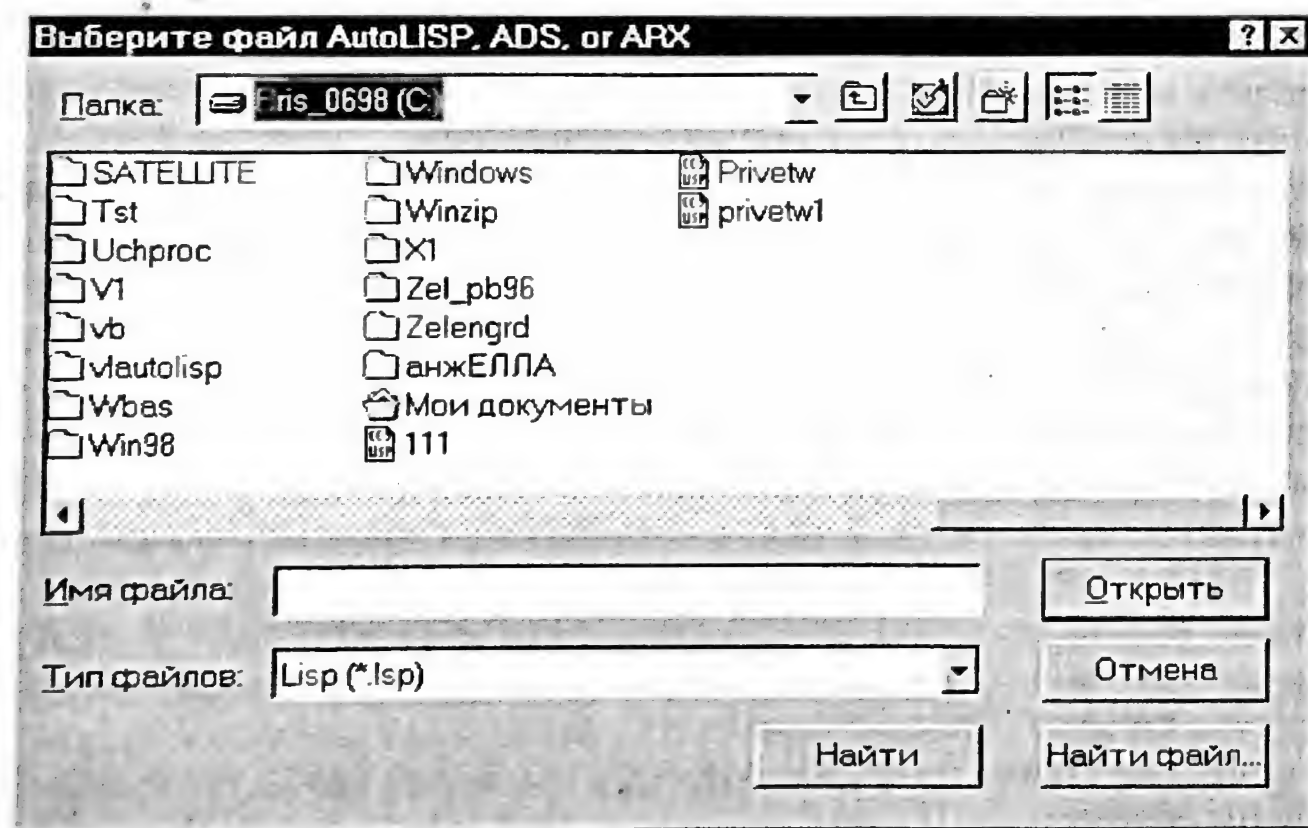


Рис. 2.21. Окно выбора файлов

В открывшемся диалоговом окне включите флажок **Сохранить список** (Save LIST). В следующий раз при выборе команды **Load Application...** (Загрузить приложение) падающего меню Tools имя файла будет внесено в диалоговое окно загрузки файлов автоматически.

Далее в этом диалоговом окне щелкните мышкой по кнопке **Файл...** (File...), появится окно выбора файлов **Выберите файл AutoLISP, ADS или ARX** (Select AutoLISP, ADS, or ARX File), показанное на рис. 2.21.

Щелкните мышкой по кнопке **Переход на один уровень вверх** (кнопка со стрелкой вверх), чтобы перейти на диск **С:**. Найдите файл Privetw и дважды щелкните по нему мышкой, появятся каталоги (папки) и файлы на диске **С:**.

После этого на экране вновь появится диалоговое окно загрузки файлов **Загрузить AutoLISP, ADS, и ARX файлы** (Load AutoLISP, ADS, and ARX Files), однако в нем уже будет файл PRIVETW.

Щелкните по кнопке **Load** (Загрузить), появится окно AutoCAD с приветствием (рис. 2.21).

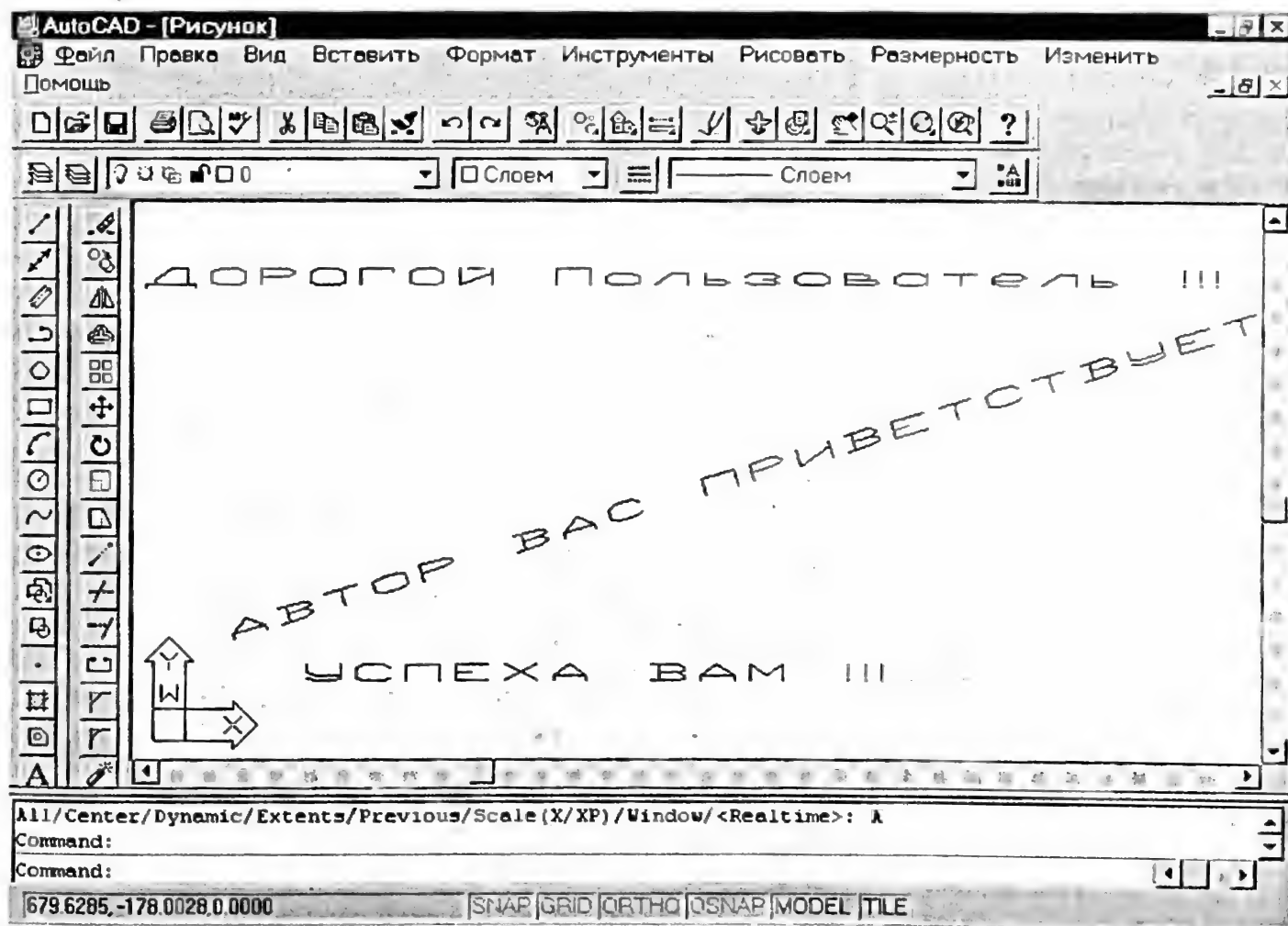


Рис. 2.22. Графическое окно системы AutoCAD

Со временем пользователь, как правило, наработывает много программ на языке AutoLISP. Чтобы в них ориентироваться, необходимо давать файлу такое оригинальное имя, которое соответствует содержанию программы. Например, имя MNKV.LSP можно дать файлу, в котором разработана программа для статистической обработки данных методом наименьших квадратов. Как можно заметить, в имени программы используются начальные буквы названия метода. Скажем, программа, предназначенная для расчета сетевого вероятностного графика, может быть названа так: SETV.LSP.

Однако, например, метод наименьших квадратов может быть предназначен для определения параметров линейного уравнения регрессии, квадратического или еще более высоких степеней. Тогда соответствующие названия файлов могут выглядеть следующим образом: MNKV_1.LSP, MNKV_2.LSP, MNKV_M.LSP.

Если, допустим, разработана программа для решения дифференциальных уравнений методом Рунге-Кутты, то можно ее назвать так: RUNG-KUT.LSP.

Желательно, чтобы число символов в названии файлов не превышало 8.

СОЗДАНИЕ ПАРАМЕТРИЧЕСКИХ ИЗОБРАЖЕНИЙ ОБЪЕКТА

Параметрическое изображение плана объекта	110
Создание параметрического изображения многоступенчатых объектов	117
Нанесение размеров на многоступенчатом объекте и ввод текста	127
Создание параметрических изображений проекций конструкций	138
Создание параметрического изображения общего вида объекта	146
Создание параметрического изображения общего вида машины	150

Эта глава посвящена вопросам разработки программ параметрического представления изображений различной сложности – от многоступенчатого объекта, ферм, узлов до общего вида объектов.

3.1. Параметрическое изображение плана объекта

3.1.1. Постановка задачи

Требуется разработать алгоритм параметрического изображения плана офиса с размещением мебели внутри него и создать соответствующую функцию для построения плана офиса с мебелью. Функция параметрического изображения плана офиса должна быть представлена в таком виде:

(OFFIS X Y LDD LD)

где X, Y – базовая точка для изображения плана; LDD – список данных для изображения двери офиса следующего вида:

' (N D2 HD), в котором N, D2 и HD – номер угла офиса, расстояние от угла офиса до двери и ширина двери соответственно;

LD – список элементов офиса с размерами (мебели)

((X1 Y1 D1 H1 "имя" U1) (X2 Y2 D2 H2 "имя" U2) ...)

где X1, Y1 – базовые координаты офиса или мебели; D1, H1 – длина и ширина офиса или мебели;

"имя", U1 – название мебели, угол установки ее.

На плане должны быть представлены упрощенные изображения мебели, ее расположение, название и размеры.

3.1.2. Выявление основных особенностей, взаимосвязей и количественных закономерностей

Сделаем набросок возможного плана офиса с мебелью (рис. 3.1).

Как можно заметить, в эскизе плана офиса преобладают фигуры типа прямоугольника. Поэтому вначале рассмотрим процесс создания изображения прямоугольника. Прямоугольник заданного размера должен быть расположен в заданной точке под заданным углом и содержать пояснительную надпись. В процессе создания изображения прямоугольника предстоит сделать следующее:

- определить точки прямоугольника относительно базовой точки с заданным углом наклона прямоугольника и заданными размерами;
- сформировать изображения прямоугольника по четырём точкам;
- определить точки начала вывода текстовой информации;
- сформировать в прямоугольнике текстовую информацию.

Допустим, имя этой функции будет **BOX**, а ее аргументы:

X, Y – координаты базовой точки прямоугольника;

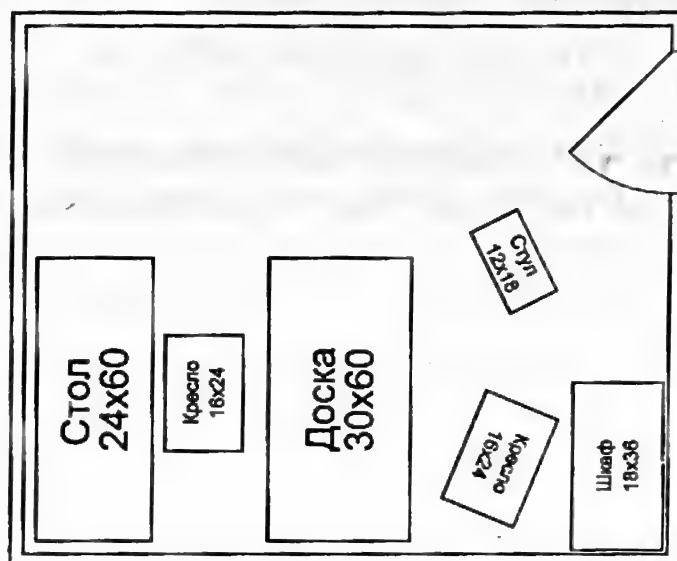


Рис. 3.1. Эскиз плана офиса с мебелью

D, H – длина и ширина прямоугольника соответственно;

"имя" – название мебели (для офиса без имени);

UG – угол наклона плана офиса или мебели в градусах.

3.1.3. Разработка последовательности действий и комплекса функций для параметрического изображения плана офиса

Для определения вершин прямоугольника используем встроенную функцию AutoLISP (**POLAR** <T1> <UR> <R>), которая определяет точку T2 путем перемещения из точки <T1> под углом <UR> в радианах на расстояние <R>. Поскольку углы наклона мебели приводятся в градусах, предварительно необходимо перевести их в радианы:

```
(SETQ UR (* PI (/ UG 180.0)))
```

Для того чтобы ввести текст в прямоугольнике (мебели), необходимо определить координаты точек ввода текста и сам текст. При этом текст вводится только для изображения мебели, то есть когда, например, индекс J>2. Этот фрагмент программы будет выглядеть следующим образом:

; определение координат точек для ввода текста и сам текст

```
(SETQ BTT (POLAR BT (+ UR (/ PI 2.7)) (/ H 2))
      BTX (POLAR BT (+ UR (/ PI 3.7)) (/ H 1.7))
      H1 (ITOA H)
      D1 (ITOA D)
      TTT "")
TTT (IF (> J 2) (STRCAT D1 "*" H1))
J (+ J 1))
```

```
; ввод текстовой информации в прямоугольник
(COMMAND "TEXT" "C" BTT (/ D 5) (+ 90 UG) TXT
      "TEXT" "C" BTX "" "" TTT)
```

Для изображения двери офиса необходимо знать координаты вершин (углов) плана офиса. Они определяются при первом использовании функции (BOX...). Остается только их запомнить. Для этого можно использовать функцию (IF...) в таком виде:

```
(IF (= J 1) (SETQ TT1 BT      ; координаты первого угла офиса
                TT2 T1      ; координаты второго угла офиса
                TT3 T2      ; координаты третьего угла офиса
                TT4 T3))    ; координаты четвертого угла офиса
```

На основании вышеизложенного приведем функцию создания изображения прямоугольника:

```
(DEFUN BOX (X Y D H TXT UG)
; определение координат вершин BT, T1, T2, T3.
  (SETQ BT (LIST X Y)      ; базовая точка офиса (мебели)
        UR (* PI (/ UG 180.0)) ; преобразование градусов в радианы
        T1 (POLAR BT (+ UR (/ PI 2)) H)
        T2 (POLAR T1 UR D)
        T3 (POLAR BT UR D)
  )
; создание изображения прямоугольника
  (COMMAND "LINE" BT T1 T2 T3 BT "")
; определение координат углов офиса TT1, TT2, TT3, TT4.
  (IF (= J 1) (SETQ TT1 BT      ; координаты первого угла офиса
                TT2 T1      ; координаты второго угла офиса
                TT3 T2      ; координаты третьего угла офиса
                TT4 T3))    ; координаты четвертого угла офиса
; определение координат точек для ввода текста
  (SETQ BTT (POLAR BT (+ UR (/ PI 2.7)) (/ H 2))
        BTX (POLAR BT (+ UR (/ PI 3.7)) (/ H 1.7))
        H1 (ITOA H)
        D1 (ITOA D)
        TTT ""
        TTT (IF (> J 2) (STRCAT D1 "*" H1))
        J (+ J 1))
; введение текстовой информации в прямоугольник
  (COMMAND "TEXT" "C" BTT (/ D 5) (+ 90 UG) TXT
        "TEXT" "C" BTX "" "" TTT)
)
```

Далее приведем пример списка элементов офиса:

```
'((24 24 144 120 "" 0) (28 28 136 112 "" 0)
```

```
(29 29 24 60 "Стол" 0) (80 29 30 60 "Доска" 0)
(145 29 18 36 "Шкаф" 0) (56 48 16 24 "Кресло" 0)
(132 100 12 18 "Стул" 206) (140 56 16 24 "Кресло" 162))
```

Первый элемент списка описывает внешнюю стену офиса, второй – внутреннюю стену офиса, последующие элементы – мебель офиса.

Теперь, используя функцию (**BOX** X Y D H "имя" U), разработаем функцию под названием **OFFICE** для создания изображения плана офиса с мебелью. Отрисовку плана офиса будем проводить в цикле. Введем несколько параметров для отрисовки двери. N – номер угла офиса, вблизи которого расположена дверь. Нумерацию углов начнем с нижнего левого угла (номер 1), следовательно левый верхний угол обозначим номером 2 и т. д. D2 – расстояние от угла офиса до двери, HD – ширина двери. Информацию о двери офиса будем представлять также в виде списка '(N D2 HD). Программа создания изображения офиса может выглядеть следующим образом:

```
(DEFUN OFFICE (X Y LDD LD) ; начало создания функции офис
  (SETVAR "CMDECHO" 0) ; отключение эха команд
  (SETVAR "BLIPMODE" 0) ; отключение изображения маркера
  (COMMAND "LIMITS" "0,0" "192,168" "ZOOM" "A")
  (SETQ N (NTH 0 LDD) ; номер угла отсчета двери
    D2 (NTH 1 LDD) ; расстояние от двери до угла
    HD (NTH 2 LDD)) ; ширина двери
  (SETQ J 1)
  ; цикл создания изображения плана офиса и мебели в нем
  (FOREACH EL LD
    (SETQ X (NTH 0 EL) ; координата по оси x
      Y (NTH 1 EL) ; координата по оси y
      D (NTH 2 EL) ; длина офиса (мебели)
      H (NTH 3 EL) ; ширина офиса (мебели)
      TXT (NTH 4 EL) ; название мебели
      U (NTH 5 EL)) ; угол установки мебели, градусы
    (IF (EQ J 1) (SETQ X1 X))
    (IF (EQ J 2) (SETQ HO (- X X1))) ; толщина стены офиса
    (BOX X Y D H TXT U) ; изображение офиса (мебели)
  )
)
```

Далее создадим фрагмент программы для изображения двери офиса. Для начала определим координаты расположения точек двери с учетом различного местоположения двери и привязкой ее к конкретному углу офиса (N). Допустим, дверь расположена в левой стене офиса. Тогда ее местоположение можно определить путем перемещения вверх от первого угла (левого нижнего под N = 1) на заданное расстояние D2.


```

; определение координат расположения точек двери
(COND ((= N 1) (SETQ TR1 (POLAR TT1 (/ PI 2) D2)
                    TR2 (POLAR TR1 (/ PI 2) HD)
                    TP1 (POLAR TR1 0 HO)
                    TP2 (POLAR TR2 0 HO)))
      ((= N 2) (SETQ TR1 (POLAR TT2 0 D2)
                    TR2 (POLAR TR1 0 HD)
                    TP1 (POLAR TR1 (* (/ 3 2.0) PI) HO)
                    TP2 (POLAR TR2 (* (/ 3 2.0) PI) HO)))
      ((= N 3) (SETQ TR1 (POLAR TT3 (* (/ 3 2.0) PI) D2)
                    TR2 (POLAR TR1 (* (/ 3 2.0) PI) HD)
                    TP1 (POLAR TR1 PI HO)
                    TP2 (POLAR TR2 PI HO)))
      ((= N 4) (SETQ TR1 (POLAR TT4 PI D2)
                    TR2 (POLAR TR1 PI HD)
                    TP1 (POLAR TR1 (/ PI 2) HO)
                    TP2 (POLAR TR2 (/ PI 2) HO))))
(SETQ TR3 (POLAR TP1 (- (/ PI 4) (* (/ PI 2) (- N 1))) HD))
; изображение двери
(COMMAND "BREAK" TR1 TR2           ; удаление части стены офиса
 "BREAK" TP1 TP2                   ; удаление части стены офиса
 "LINE" TP1 TR1 ""                 ; изображение торца стен офиса
 "LINE" TP2 TR2 ""                 ; изображение торца стен офиса
 "LINE" TP1 TR3 ""                 ; изображение открытой двери
 "ARC" TR3 "END" TP2 TR2)          ; изображение следа двери

```

В окончательном виде программа создания параметрического изображения плана офиса представлена ниже.

3.1.4. Программа (функция) параметрического изображения плана офиса

```

; *****
; (OFFICE X Y LDD LD) - функция параметрического построения
; плана офиса
; А р г у м е н т ы   ф у н к ц и и:
; X,Y - базовая точка для изображения плана (X Y)
; LDD - список элементов для изображения двери офиса
; ' (N D2 HD)
; N - номер угла офиса
; D2 - расстояние от угла офиса до двери
; HD - ширина двери
; LD - список элементов офиса с размерами (мебели)
; ((X1 Y1 D1 H1 "имя" U1) (X2 Y2 D2 H2 "имя" U2)...)

```

```
; X1, Y1 - базовые координаты офиса или мебели
; D1, H1 - длина и ширина офиса или мебели T1
; "имя", U1 - название мебели, угол ее установки
; (BOX X Y D H "имя" U) - функция изображения мебели (плана)
; X, Y - базовые координаты, D - длина, H - ширина, "имя",
; U - название и угол установки плана офиса или мебели
; Пример вызова функции (OFFICE BT LD)
; (OFFICE 24 24 ' ((24 24 144 120 "" "") (28 28 136 112
; "" "") (29 29 24 60 "Стол" "") ...)
; *****
(DEFUN BOX (X Y D H TXT UG)
; определение координат вершин прямоугольника
  (SETQ BT (LIST X Y) ; базовая точка офиса (мебели)
    UR (* PI (/ UG 180.0)) ; преобразование градусов в радианы
    T1 (POLAR BT (+ UR (/ PI 2)) H)
    T2 (POLAR T1 UR D)
    T3 (POLAR BT UR D)
  )
; создание изображения прямоугольника
  (COMMAND "LINE" BT T1 T2 T3 BT "")
  (IF (= J 1) (SETQ TT1 BT ; координаты первого угла офиса
    TT2 T1 ; координаты второго угла офиса
    TT3 T2 ; координаты третьего угла офиса
    TT4 T3) ; координаты четвертого угла офиса
  )
; определение координат точек для ввода текста
  (SETQ BTT (POLAR BT (+ UR (/ PI 2.7)) (/ H 2))
    BTX (POLAR BT (+ UR (/ PI 3.7)) (/ H 1.7))
    H1 (ITOA H)
    D1 (ITOA D)
    TTT ""
    TTT (IF (> J 2) (STRCAT D1 "*" H1))
    J (+ J 1)
  )
; введение текстовой информации в прямоугольник
  (COMMAND "TEXT" "C" BTT (/ D 5) (+ 90 UG) TXT
    "TEXT" "C" BTX "" "" TTT)
)
(DEFUN OFFICE (X Y LDD LD) ; создание плана офиса
  (SETVAR "CMDECHO" 0) ; отключение эха команд
  (SETVAR "BLIPMODE" 0) ; отключение изображения маркера
  (COMMAND "LIMITS" "0,0" "192,168" "ZOOM" "A")
  (SETQ N (NTH 0 LDD) ; номер угла отсчета двери
    D2 (NTH 1 LDD) ; расстояние от двери до угла
```

```

      HD (NTH 2 LDD))           ; ширина двери
(SETQ J 1)
; цикл изображения плана офиса и мебели в нем
(FOREACH EL LD
  (SETQ X (NTH 0 EL))           ; координата по оси X
    Y (NTH 1 EL)               ; координата по оси Y
    D (NTH 2 EL)               ; длина офиса (мебели)
    H (NTH 3 EL)               ; ширина офиса (мебели)
    TXT (NTH 4 EL)             ; название мебели
    U (NTH 5 EL))              ; угол установки мебели, градусы
  (IF (EQ J 1) (SETQ X1 X))
  (IF (EQ J 2) (SETQ HO (- X X1))) ; толщина стены офиса
  (BOX X Y D H TXT U)          ; изображение офиса (мебели)
)
; определение координат расположения точек двери
(COND ((= N 1) (SETQ TR1 (POLAR TT1 (/ PI 2) D2)
      TR2 (POLAR TR1 (/ PI 2) HD)
      TP1 (POLAR TR1 0 HO)
      TP2 (POLAR TR2 0 HO)))
  ((= N 2) (SETQ TR1 (POLAR TT2 0 D2)
      TR2 (POLAR TR1 0 HD)
      TP1 (POLAR TR1 (* (/ 3 2.0) PI) HO)
      TP2 (POLAR TR2 (* (/ 3 2.0) PI) HO)))
  ((= N 3) (SETQ TR1 (POLAR TT3 (* (/ 3 2.0) PI) D2)
      TR2 (POLAR TR1 (* (/ 3 2.0) PI) HD)
      TP1 (POLAR TR1 PI HO)
      TP2 (POLAR TR2 PI HO)))
  ((= N 4) (SETQ TR1 (POLAR TT4 PI D2)
      TR2 (POLAR TR1 PI HD)
      TP1 (POLAR TR1 (/ PI 2) HO)
      TP2 (POLAR TR2 (/ PI 2) HO))))
(SETQ TR3 (POLAR TP1 (- (/ PI 4) (* (/ PI 2) (- N 1))) HD))
; изображение двери
(COMMAND "BREAK" TR1 TR2       ; удаление части стенки офиса
  "BREAK" TP1 TP2             ; удаление части стенки офиса
  "LINE" TP1 TR1 ""           ; соединение кончиков стен офиса
  "LINE" TP2 TR2 ""           ; соединение кончиков стен офиса
  "LINE" TP1 TR3 ""           ; изображение открытой двери
  "ARC" TR3 "END" TP2 TR2)    ; изображение следа двери
)
(OFFICE 24 24 '(3 10 30) '((24 24 144 120 "" 0) (28 28 136 112 "" 0) (29 29
24 60 "Стол" 0) (80 29 30 60 "Доска" 0) (145 29 18 36 "Шкаф" 0) (56 48 16 24
"Кресло" 0) (132 100 12 18 "Стул" 206) (140 56 16 24 "Кресло" 162)))

```

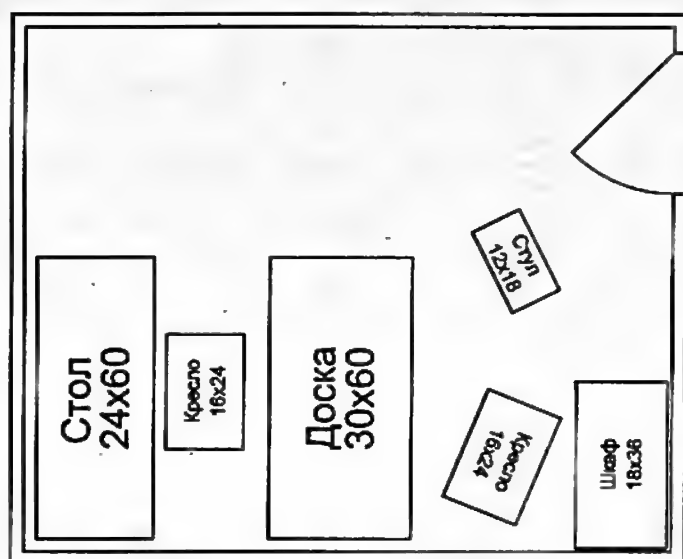


Рис. 3.2. Изображение плана офиса функцией (OFFICE ...)

После загрузки программы и ее выполнения будет создано изображение плана офиса с мебелью в графическом окне AutoCAD (рис. 3.2).

3.2. Создание параметрического изображения многоступенчатых объектов

3.2.1. Постановка задачи

Создать различные функции на AutoLISP для построения изображения многоступенчатого объекта (колонны, ракеты, вала и т. д.) с заданным числом ступеней – N и заданными размерами каждой ступени: ширина (диаметр) ступени – D ; длина ступени – L .

3.2.2. Выявление основных особенностей, взаимосвязей и количественных закономерностей

Вначале подготовим эскиз простейшего трехступенчатого объекта (рис. 3.3) и постараемся выявить особенности создания его изображения.

Изображение трехступенчатого объекта можно выполнить различными способами, например, отобразить первую ступень объекта, затем вторую и т. д. или нарисовать общий контур объекта, затем четко выделить ступени. Можно начать рисование слева, справа, снизу, сверху...

Выделим некоторые особенности прорисовки многоступенчатого объекта:

- во-первых, желательно рисовать объект таким образом, чтобы впоследствии не пришлось прорисовывать один и тот же элемент дважды;

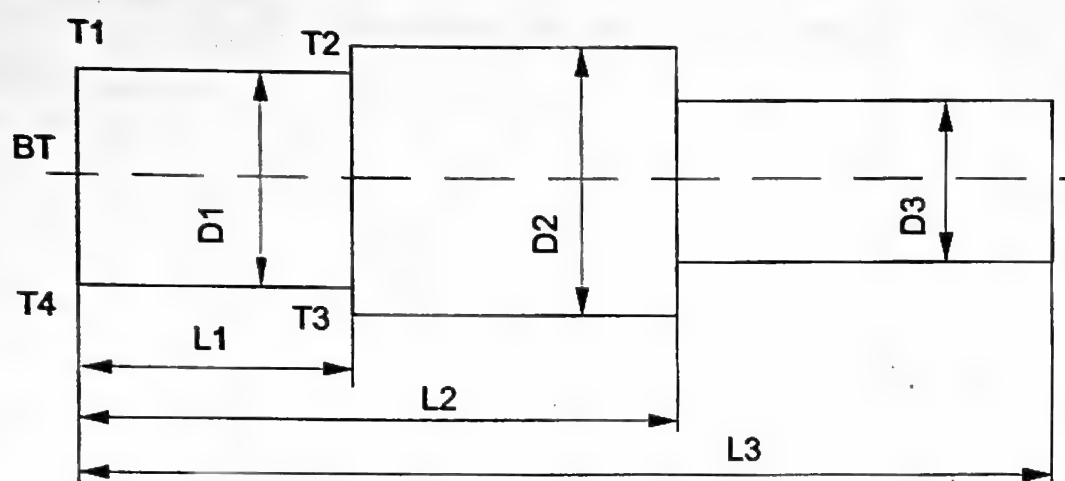


Рис. 3.3. Трехступенчатый объект

- во-вторых, следует выбрать такую технологию прорисовки объекта, чтобы ее можно было использовать при прорисовке объекта с любым числом ступеней.

Логичнее начать прорисовку объекта слева направо относительно некоторой базовой точки (ВТ), например, лежащей на левом торце объекта на осевой линии (рис. 3.3).

Из анализируемого чертежа объекта видно, что первую его ступень можно изобразить ломаной линией, проходящей через четыре точки, принадлежащие первой ступени: Т1, Т2, Т3, Т4. Координаты этих точек легко определить, если известны координаты базовой точки – ВТ, ширина (диаметр) объекта – D и длина первой ступени – L .

3.2.3. Разработка последовательности действий и комплекса функций для параметрического изображения многоступенчатого объекта

Для создания изображения любой ступени достаточно знать базовую точку ступени объекта, размеры ступени и уметь определять координаты точек каждой ступени объекта. Координаты следующей базовой точки ступени объекта можно найти путем перемещения от базовой точки предыдущей ступени объекта на величину L , равную длине ступени объекта, и т.д.

Координаты точек ступени объекта можно определить, если переместиться, например, от базовой точки ступени объекта вверх и вниз на расстояние, равное половине ширины (диаметра) ступени. В результате получим координаты точек Т1 и Т4. Из этих точек следует переместиться вдоль объекта на расстояние, равное длине ступени объекта – получим точки Т2 и Т3. Затем построим изображение ступени объекта, используя

функцию (**COMMAND...**) и команды графического редактора AutoCAD **"PLINE"**:

```
(COMMAND "PLINE" BT "W" 1 1 T1 T2 T3 T4 BT " ")
```

где **"PLINE"** – команда графического редактора для изображения полилинии, проходящей через несколько точек, с заданной шириной;

"W" (Width) – ширина.

Первая цифра определяет ширину линии в начале, вторая цифра – в конце полилинии.

С помощью цикла можно последовательно изобразить все имеющиеся ступени. Создание изображения многоступенчатого объекта можно осуществить как в диалоговом режиме, так и путем обращения к функции с заданными параметрами.

Разработку программы начнем с оформления ввода исходных данных в диалоговом режиме. Будем запрашивать число ступеней объекта (**N**).

На языке AutoLISP этот запрос будет выглядеть следующим образом:

```
(SETQ N (GETINT "\n Введите число ступеней объекта N = "))
```

где (**GETINT...**) – встроенная функция, которая обеспечивает ввод целого числа (**GET INTegeR** – получить целое число).

Подсказка оформляется как текстовая константа (в кавычках). Ее длина не должна превышать 100 символов.

(**SETQ...**) – встроенная функция, которая обеспечивает присвоение переменной **N** значения функции (**GETINT...**), то есть числа ступеней объекта. После выполнения функции (**SETQ...**), если, например, было введено число ступеней, равное 4, получим результат – значение переменной **N = 4**.

Символы **\n** в подсказке в текстовой константе являются управляющими – они обеспечивают перевод подсказки на новую строку.

Имя переменной может состоять из латинских букв, цифр, специальных символов, кроме **() . ' " ;** и должно начинаться с буквы. В LISP переменные и константы называются *атомами*. К *константам* относятся числа, целые и вещественные; строки символов, заключенные в кавычки (текстовые константы); логические константы (**T** – истина, **NIL** – ложь).

AutoLISP содержит множество различных функций ввода, в том числе:

(**GETREAL...**) – ввод вещественного числа;

(**GETSTRING...**) – ввод строки текста;

(**GETPOINT...**) – ввод координат точки;

(**GETDIST...**) – ввод расстояния и другие.

Базовую точку определим с помощью координат **x** и **y**.

```
(SETQ XT (GETREAL "\n Введите координату x базовой точки XT = ")
      YT (GETREAL "\n Введите координату y базовой точки YT = ")
      BT (LIST XT YT))
```

Точка в AutoLISP определяется как список координат X и Y.

Функция (LIST XT YT) задает базовую точку BT. Запрос базовой точки можно выполнить с помощью функции (GETPOINT...), которая в нашей задаче записывается следующим образом:

```
(SETQ BT (GETPOINT "\n Введите базовую точку BT "))
```

В ответ на запрос вводятся сразу две координаты: X и Y, без пробелов через запятую, например, 7,15.

Разработка цикла для ввода размеров каждой ступени и создание ее изображения могут быть выполнены следующими различными функциями:

- (REPEAT <число> <выражение>...) – вычисляет выражения заданное число раз;
- (WHILE <условие> <выражение>...) – вычисляет выражения до тех пор, пока выполняется заданное условие (выражение-предикат);
- (FOREACH <переменная> <список> <выражение>...) – вычисляет выражения для всех элементов списка. Сначала присваивает переменной значение первого элемента списка и вычисляет все входящие в функцию выражения, затем присваивает переменной значение второго элемента списка и снова вычисляет все выражения и т. д., включая последний элемент списка.

Используем в нашей задаче простейшую функцию формирования цикла (REPEAT <число> <выражение>...). В одном цикле будем запрашивать ввод ширины (диаметра) ступени объекта – D и длины ступени объекта – L:

```
(REPEAT N
  (SETQ D (GETREAL "\n Введите диаметр ступени объекта D = ")
    L (GETREAL "\n Введите длину ступени объекта L = ")
    R (* D 0.5))
)
```

В этом же цикле будем определять точки ступени и формировать ее изображение. Для определения точек используем функцию

```
(POLAR <точка> <угол> <расстояние>)
```

Углы в AutoLISP по умолчанию отсчитываются от горизонтальной оси против часовой стрелки. Определение точек одной ступени будет выглядеть так:

```
(SETQ T1 (POLAR BT (/ PI 2) R)
      T2 (POLAR T1 0 L)
      T4 (POLAR BT (* 1.5 PI) R)
      T3 (POLAR T4 0 L)
)
```

Создание изображения объекта выполняется с помощью функции AutoLISP:

```
(COMMAND "<команда AutoCAD>" ...[аргументы]...[ключи]...)
```

Для формирования полилинии используется команда "PLINE", которая позволяет отрисовывать элементы чертежа, состоящие из прямолинейных и дуговых сегментов (с указанием ширины). Эта команда может иметь различные ключи в зависимости от режима изображения линий или режима изображения дуг.

В режиме изображения линий используются следующие ключи:

- "A" (Arc) – переход в режим дуги;
- "C" (Close) – замкнуть отрезком;
- "L" (Length) – суммарная длина линий.

В режиме изображения дуг используются следующие ключи:

- "A" (Angle) – центральный угол;
- "CE" (CEnter point) – центр дуги;
- "CL" (CLose) – замкнуть дугой;
- "D" (Direction) – направление;
- "L" (Line) – переход в режим линий;
- "R" (Radius) – радиус дуги;
- "S" (Second point) – вторая точка;
- "U" (Undo) – отмена последней линии.

Ключи для изменения ширины полилинии:

- "H" (Half-width) – полуширина;
- "W" (Width) – ширина линии.

Создание изображения одной ступени будет выглядеть так:

```
(COMMAND "PLINE" BT "W" 0.1 0.1 T1 T2 T3 T4 "C")
(SETQ BT (POLAR BT 0 L))
```

Ниже представлена функция (MNOGOSTD) создания изображения многоступенчатого объекта в диалоговом режиме.

3.2.4. Программа (функция) параметрического изображения многоступенчатого объекта в диалоговом режиме

```
; *****
; (MNOGOSTD) – функция создания параметрического изображения
; многоступенчатого объекта в диалоговом режиме
; *****
(DEFUN MNOGOSTD () ; начало создания функции;
  (SETVAR "CMDECHO" 0) ; отключение эха команд
  (SETVAR "BLIPMODE" 0) ; отключение изображения маркера
  (COMMAND "LIMITS" "0,0" "297,210" "ZOOM" "A")
```

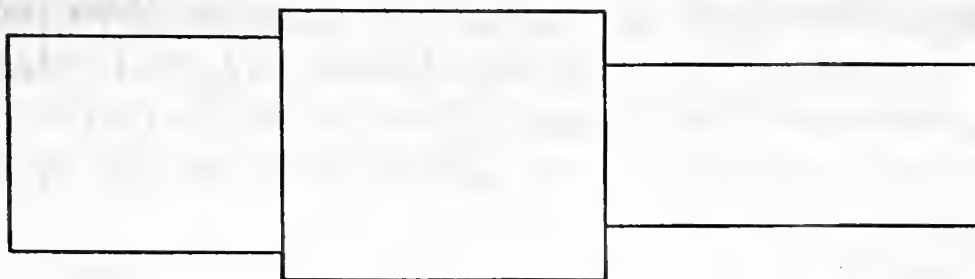



Рис. 3.4. Параметрическое изображение многоступенчатого объекта, созданное при помощи функции (MNOGOSTD)

```
(TEXTSCR)                                ; переход на текстовый экран
(SETQ N (GETINT "\n Введите число ступеней объекта N = "))
  XT (GETREAL "\n Введите координату x базовой точки XT = ")
  YT (GETREAL "\n Введите координату y базовой точки YT = ")
  BT (LIST XT YT))
(REPEAT N
  (SETQ D (GETREAL "\n Введите диаметр ступени объекта D = ")
    L (GETREAL "\n Введите длину ступени объекта L = ")
    R (* D 0.5))
  ; определение характерных точек ступени объекта
  (SETQ T1 (POLAR BT (/ PI 2) R)
    T2 (POLAR T1 0 L)
    T4 (POLAR BT (* 1.5 PI) R)
    T3 (POLAR T4 0 L) )
  (COMMAND "PLINE" BT "W" 0.1 0.1 T1 T2 T3 T4 "C")
  (SETQ BT (POLAR BT 0 L)))
)
(MNOGOSTD)                                ; вызов функции
```

Для создания многоступенчатого объекта, изображенного на рис. 3.4, необходимо ввести следующие значения в ответ на соответствующие запросы:

Введите число ступеней объекта N = 3
 Введите координату x базовой точки XT = 20
 Введите координату y базовой точки YT = 160
 Введите диаметр ступени объекта D = 40
 Введите длину ступени объекта L = 50
 Введите диаметр ступени объекта D = 50
 Введите длину ступени объекта L = 60
 Введите диаметр ступени объекта D = 30
 Введите длину ступени объекта L = 70

Разработаем теперь функцию (MNOGOSTP...), исходные данные которой вводятся через аргументы функции. В качестве аргументов возьмем базовую точку объекта BT и список пар LD. Каждая пара представляет собой список и включает диаметр D и длину ступени объекта L. Создание изображения

объекта также выполним в цикле, только в качестве функции цикла используем функцию (**FOREACH...**).

```
(FOREACH <переменная> <список> <выражение>...)
```

Допустим, имя переменной **EL**. Вначале присвоим переменной **EL** значение первой пары списка **LD** – параметры первой ступени объекта и вычислим все последующие выражения, входящие в функцию (**FOREACH...**). Затем присвоим переменной **EL** значение второй пары списка и т. д., пока не будут исчерпаны значения, определяющие размеры всех ступеней объекта.

Чтобы начать создание изображения ступени объекта, необходимо выделить из списка **'(D L)** значения ширины (диаметра) и длины ступени объекта. Для этого можно использовать две встроенные функции – (**CAR...**) и (**CDR...**). Функция (**CAR EL**) выделяет из списка первый элемент – ширину (диаметр) ступени; функция (**CDR EL**) – хвост списка, то есть список без первого элемента **'(L)**. Используя функцию (**CAR '(L)**), можно выделить из списка первый элемент – длину ступени. Так как эта процедура очень часто используется в AutoLISP, для упрощения ее записи применяется выражение (**CADR EL**), которое позволяет получить второй элемент списка. Диаметр и длина ступени могут быть определены как

```
(SETQ D (CAR EL)
      L (CADR EL))
```

Для получения элементов списка можно использовать и функцию (**NTH...**), тогда размеры ступени определяются следующим образом:

```
(SETQ D (NTH 0 EL)
      L (NTH 1 EL))
```

Для создания изображения многоступенчатого объекта так же, как и в функции (**MNOGOSTD**), будем использовать команду AutoCAD **"PLINE"**.

3.2.5. Программа (функция) параметрического изображения многоступенчатого объекта

```
;*****
; (MNOGOSTP BT LD) - функция создания изображения
; многоступенчатого объекта с заданными параметрами
; А р г у м е н т ы   ф у н к ц и и :
; BT - базовая точка для изображения объекта (X Y)
; LD - список пар диаметров и длин ступеней объекта
; ((D1 L1) (D2 L2) . . . )
; Пример вызова функции (MNOGOSTP BT LD)
; (MNOGOSTP '(10 90) '((80 70) (90 30) (70 40) (50 70)))
;*****
```

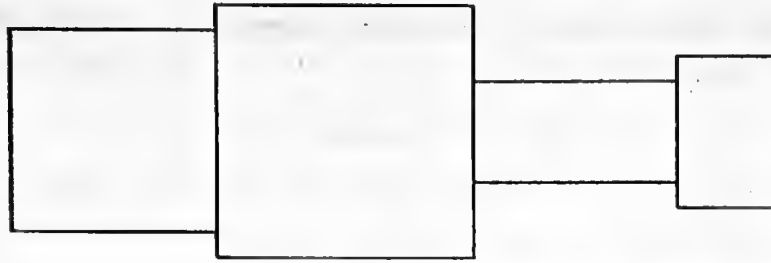


Рис. 3.5. Параметрическое изображение многоступенчатого объекта, созданное при помощи функции (MNOGOSTP...)

```
(DEFUN MNOGOSTP (BT LD) ; начало создания функции
  (SETVAR "CMDECHO" 0) ; отключение эха команд
  (SETVAR "BLIPMODE" 0) ; отключение изображения маркера
  ; установка формата чертежа
  (COMMAND "LIMITS" "0,0" "297,210" "ZOOM" "A")
  (FOREACH EL LD ; выделение элемента из списка
    (SETQ D (CAR EL) ; диаметра ступени объекта
      L (CADR EL) ; длины ступени объекта
      R (* D 0.5))
    ; определение характерных точек ступени объекта вида
    (SETQ T1 (POLAR BT (/ PI 2) R)
      T2 (POLAR T1 0 L)
      T4 (POLAR BT (* 1.5 PI) R)
      T3 (POLAR T4 0 L) )
    ; создание изображения ступени объекта
    (COMMAND "PLINE" BT "W" 0.1 0.1 T1 T2 T3 T4 "C")
    (SETQ BT (POLAR BT 0 L))
  )
  (MNOGOSTP '(10 160) '((40 40) (50 50) (20 40) (30 20)))
```

Результатом выполнения функции с заданными параметрами будет изображение многоступенчатого объекта, приведенное на рис. 3.5.

Однако без осевой линии чертеж объекта будет неполным. Отобразить осевую линию можно с помощью следующих команд AutoCAD:

```
(COMMAND "LAYER" "N" "OSI" "L" "CENTER" "OSI" "S" "OSI" ""
  "LINE" (POLAR BP PI 5) (POLAR BT 0 5))
```

Как правило, осевые линии отображают на отдельном слое. Назовем этот слой "OSI" (по умолчанию используется слой "0"). Для управления слоями существует команда "LAYER" графического редактора AutoCAD. Ключ **N** (**N**ew – новый) указывает на создание нового слоя; ключ **L** (**L**type – тип линии) определяет тип линий; ключ **S** (**S**et – установить) устанавливает текущий слой.

Чтобы задать типы линий, необходимые для получения конструкторской документации, используют следующие ключи: "CONTINUOUS" (непрерывный), "CENTER" (осевой), "DASHED" (пунктирный), "DASHDOT" (штрихпунктирный) и другие.

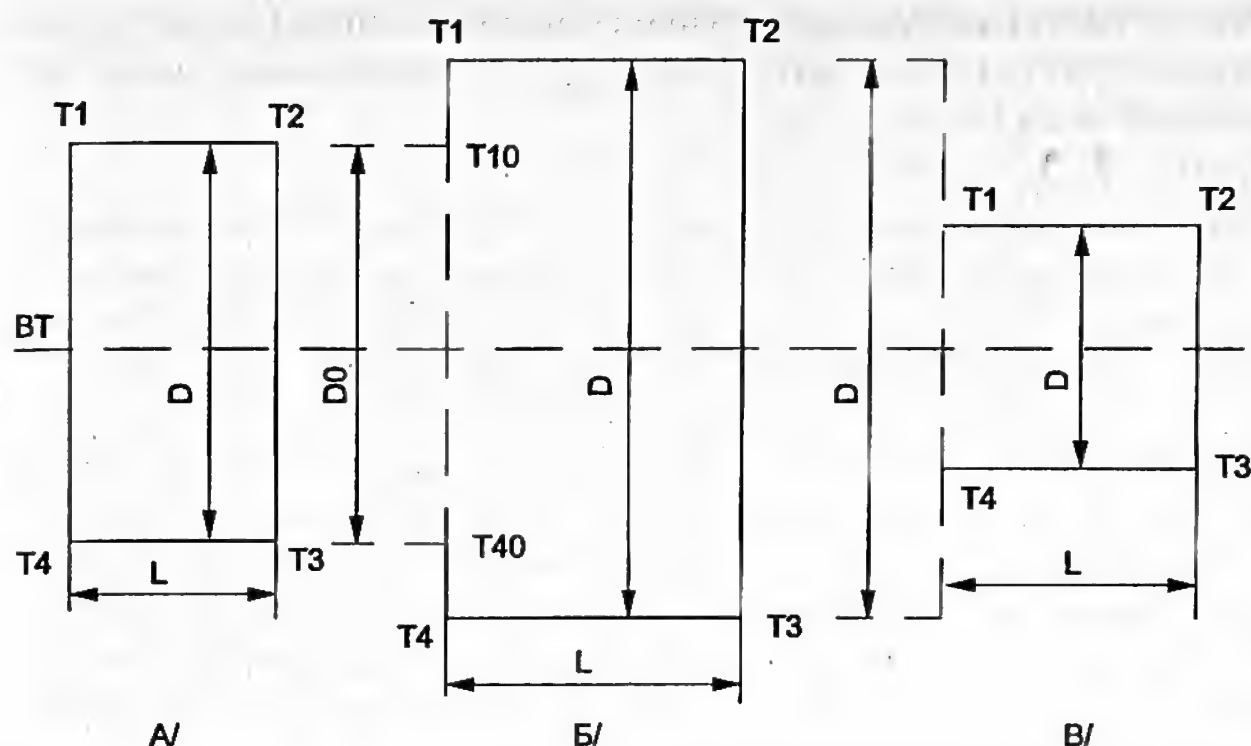


Рис. 3.6. Варианты графического изображения ступеней объекта: а) первой ступени объекта; б) ступени объекта, ширина (диаметр) которой больше ширины (диаметра) предыдущей ступени; в) ступени объекта, ширина (диаметр) которой меньше ширины (диаметра) предыдущей ступени

Двойные кавычки "" в функции (**COMMAND...**) соответствуют нажатию клавиши **Enter**. Команда "**LINE**" обеспечивает рисование осевой линии. Начало этой линии находится в точке, заданной функцией (**POLAR BP PI 5**). Конец осевой линии задается функцией (**POLAR BT 0 5**).

Напомним, что функция (**POLAR...**) формирует точку путем перемещения от заданной точки в заданном направлении на заданное расстояние. Число 5 – это расстояние перемещения от исходной точки; BP – базовая точка, определяется перед циклом:

```
(SETQ BP BT)
```

При использовании одного прохода пишущего устройства для изображения объекта возможны три способа отрисовки его ступеней (рис. 3.6).

Можно разработать функцию, которая обеспечит создание изображения многоступенчатого объекта без повторного рисования некоторых совпадающих частей. На рис. 3.6 приведены возможные варианты графического изображения ступеней объекта.

Для создания изображения первой ступени объекта можно использовать выражение вида (см. рис. 3.6, а):

```
(COMMAND "PLINE" BT "W" 1 1 T1 T2 T3 T4 BT " ")
```

Для создания изображения ступени объекта, ширина (диаметр) которой больше ширины (диаметра) предыдущей ступени, можно использовать выражение вида (см. рис. 3.6, б):

```
(COMMAND "PLINE" T10 "W" 1 1 T1 T2 T3 T4 T40 " ")
```


Для создания изображения ступени объекта, ширина (диаметр) которой меньше ширины (диаметра) предыдущей ступени, можно использовать выражение вида (см. рис. 3.6, в):

```
(COMMAND "PLINE" T1 "W" 1 1 T2 T3 T4 " ")
```

Внимательный анализ графических изображений, представленных на рис. 3.6, показывает, что каждое из них можно построить с помощью второго варианта. В первом случае (см. рис. 3.6, а) точки T10 и T40 совпадают с базовой точкой BT, в третьем случае (см. рис. 3.6, в) точка T10 совпадает с точкой T1, а точка T40 – с точкой T4. Если диаметр, на концах которого лежат точки T10 и T40, обозначить DO, то в первом случае его величина будет равна 0, во втором – диаметру предыдущей ступени, в третьем – диаметру изображаемой ступени объекта. Точки T10 и T40 определяются путем перемещения от базовой точки строящейся ступени объекта вверх и вниз на величину, равную половине ширины (диаметра) DO, соответственно. Создание графического изображения объекта можно организовать с помощью цикла.

Для определения характерных точек ступени объекта T10, T1, T40, T4, T2, T3 используется функция (LIST...), которая формирует точку по двум координатам X и Y.

Построим изображение многоступенчатого объекта с осевой линией. Сформируем также список точек, через которые пройдут выносные линии для указания размеров ширины (диаметра) ступени.

Функция (MNOGOSTPM...) удовлетворяет всем вышеперечисленным требованиям:

```
; *****
; (MNOGOSTPM X Y LD) - функция создания параметрического
; изображения многоступенчатого объекта
; А р г у м е н т ы   ф у н к ц и и:
; X - координата x базовой точки ступени
; Y - координата y базовой точки ступени
; LD - список, включающий ширину (диаметр) и длину каждой ступени
; (STUP) - функция создания изображения ступени объекта
; Пример вызова функции (MNOGOSTPM X Y LD):
; (MNOGOSTPM 50 100 '((80 70) (90 50) (70 100) (50 70)))
; *****
(DEFUN STUP ()
  (SETQ T10 (LIST X (+ Y RO))
    T1 (LIST X (+ Y R ))
    T40 (LIST X (- Y RO))
    T4 (LIST X (- Y R ))
    X (+ X L)
    T2 (LIST X (+ Y R ))
    T3 (LIST X (- Y R ))))
```

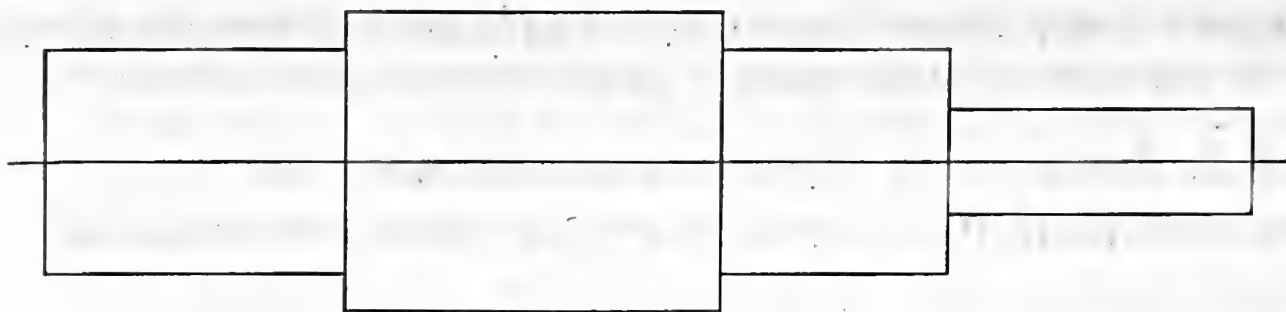


Рис. 3.7. Параметрическое изображение многоступенчатого объекта, созданное при помощи функции (MNOGOSTPM...)

```
(COMMAND "PLINE" T10 "W" 0.1 0.1 T1 T2 T3 T4 T40 "")
)
(DEFUN MNOGOSTPM (X Y LD) ; начало описания функции
  (SETVAR "CMDECHO" 0 ) ; отключение эха команд
  (SETVAR "BLIPMODE" 0 ) ; отключение изображения маркеров
  (COMMAND "LIMITS" "0,0" "297,210" "ZOOM" "A") ; установка поля
  (SETQ BP (LIST (- X 5) Y) ; начало осевой линии
    LT2T3 '() ; начало создания списка
    RO 0) ; начальное значение RO
  (FOREACH EL LD ; цикл построения ступеней
    (SETQ D (CAR EL) ; определение диаметра ступени
      L (CADR EL) ; определение длины ступени
      R (/ D 2)) ; определение радиуса ступени
    (IF (< R RO) (SETQ RO R)) ; определение RO для 1-й ступени
    (STUP) ; изображение 1-й ступени
    (SETQ RO R) ; подготовка к последнему построению
  )
  (SETQ BT (LIST (+ X 5) Y)) ; конец осевой линии
  (COMMAND "LAYER" "N" "OSI" "L" "CENTER" "OSI" "S" "OSI"
    "" "LINE" BP BT "")
  (PRIN1)
)
(MNOGOSTPM 10 60 '((30 40) (40 50) (30 30) (15 40))) ; вызов функции
```

В результате выполнения функции (MNOGOSTPM...) получим изображение многоступенчатого объекта (рис. 3.7).

3.3. Нанесение размеров на многоступенчатом объекте и ввод текста

3.3.1. Постановка задачи

Допустим, известна функция создания изображения многоступенчатого объекта с заданным числом ступеней, заданными диаметрами и длиной

каждой ступени. Требуется разработать функцию, обеспечивающую нанесение размеров (образмеривание) на многоступенчатом объекте.

3.3.2. Выявление основных особенностей, взаимосвязей и количественных закономерностей

Изображение размеров включает следующие составляющие:

- *размерную линию* – линию со стрелками (засечками), проведенную параллельно измерению;
- *выносные линии* – линии, обеспечивающие вынос размерной линии вне образмериваемого элемента объекта;
- *размерный текст* – текстовую строку, содержащую размер.

Предположим, что размерная линия для указания размера диаметра должна отстоять от правой границы соответствующей ступени объекта на 10 ед. влево.

Указывать размеры длины ступеней объекта будем слева направо, так, как показано на рис. 3.8.

Для указания размера ширины (диаметра) ступени объекта необходимо определить:

- положение размерной линии: вертикальное или горизонтальное (в нашей задаче эта линия должна быть вертикальной);
- координаты точек начала выносных линий (в нашей задаче такими точками для первой ступени объекта могут быть T12 и T13);
- координаты точки, через которую пройдет размерная линия (назовем эту точку T1R). Координаты данной точки можно определить, зная координаты точки T12 и сместившись относительно этой точки влево по горизонтали на заданное число единиц.

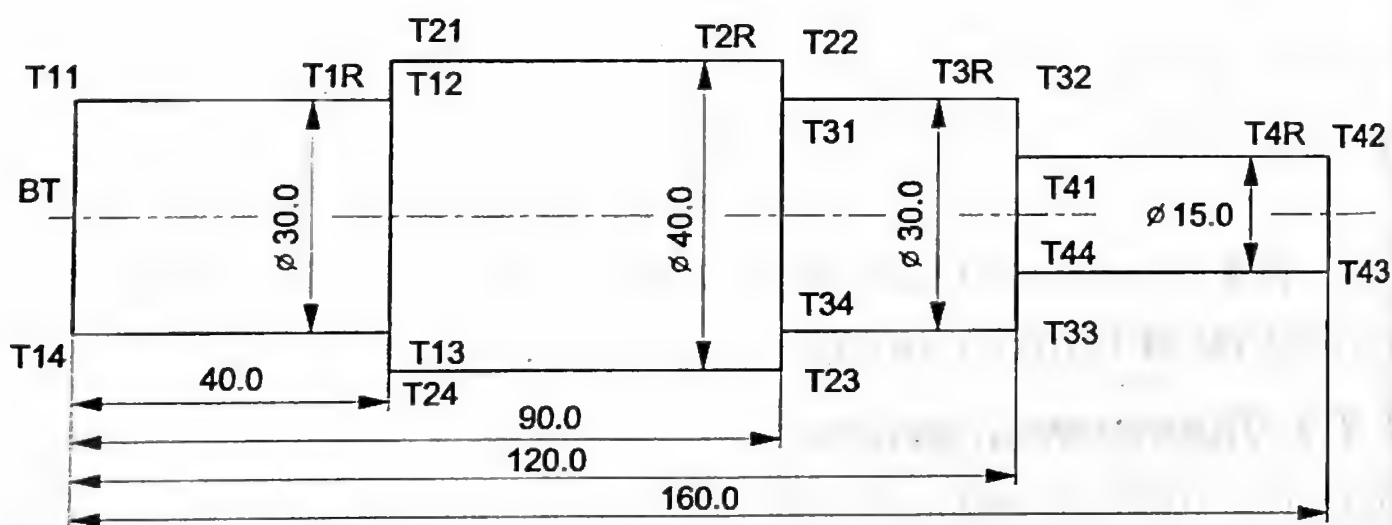


Рис. 3.8. Изображение многоступенчатого объекта с размерами и обозначение точек

Проставить размер диаметра необходимо на каждой ступени объекта. Для этого можно сформировать цикл и использовать список пар точек правого торца каждой ступени объекта. Для объекта, представленного на рис. 3.8, список пар точек будет выглядеть так:

((T12 T13) (T22 T23) (T32 T33) (T42 T43))

Чтобы проставить размер длины ступени объекта так же, как и диаметр объекта, необходимо определить:

- положение размерной линии (в нашей задаче – горизонтальное);
- координаты точек выносных линий (в нашей задаче первой точкой для всех ступеней объекта может служить базовая точка объекта ВР, второй – базовая точка следующей ступени объекта ВТ или вторая (нижняя) точка из пары точек);
- точку, через которую будет проходить размерная линия (назовем ее T1R).

Для того чтобы проставить длину ступеней объекта, необходимо сформировать цикл.

3.3.3. Разработка последовательности действий и программы нанесения размеров на многоступенчатом объекте

Формирование размеров многоступенчатого объекта предполагает выполнение следующих действий:

- формирование цикла для указания размера диаметра ступени объекта;
- выделение необходимой пары точек начала выносных линий из списка пар точек для указания диаметра ступени;
- выделение из пары точки начала первой выносной линии;
- выделение из пары точки начала второй выносной линии;
- определение точки положения размерной линии;
- указание размера диаметра ступени объекта;
- определение точки положения линии для указания размера длины ступени;
- выделение точки начала для второй выносной линии;
- указание размера длины первой ступени;
- формирование цикла для указания размеров длины.

Определим список пар точек для указания размеров. В языке AutoLISP есть встроенная функция создания списка (**LIST...**) (список, напомним, – это упорядоченная последовательность элементов, заключенных в круглые скобки).

(LIST <элемент> <элемент>...)

Элементами могут быть переменные, числа, строковые и логические константы (их называют *атомами*) и списки. Элементы разделяются пробелами. Например, список '(A B '(C D) E) состоит из четырех элементов, где третий элемент представляет собой список из двух элементов.

Сформируем список пар точек начала выносных линий для каждой ступени объекта и назовем этот список LT2T3. Формирование указанного списка можно начать следующим образом:

```
(SETQ LT2T3 '()
      LT2T3 (CONS (LIST T12 T13) LT2T3))
```

где функция (LIST...) создает список из двух точек T12 и T13. Функция (CONS...) формирует список из пар точек.

Функция (CONS...) добавляет новый элемент в начало списка:

```
(CONS <новый элемент> <список>)
```

В результате выполнения функции (SETQ...) переменной LT2T3 будет соответствовать список вида '('(T12 T13)).

Список пар точек для двух ступеней объекта можно построить так (см. рис. 3.8):

```
(SETQ LT2T3 (CONS (LIST T22 T23) LT2T3))
```

В начало прежнего списка добавляется список из двух точек '(T22 T23). В результате переменная LT2T3 будет содержать список '('(T22 T23) '(T12 T13)).

Если следующая ступень объекта – ступень меньшей ширины (диаметра), то расширение списка будет выглядеть так (см. рис. 3.8):

```
(SETQ LT2T3 (CONS (LIST T32 T33) LT2T3))
```

Для того, чтобы формирование списка пар точек можно было осуществить в цикле, достаточно использовать для обозначения точек вторую цифру (см. рис. 3.8).

В AutoCAD имеется множество видов команд для нанесения размеров. Кроме того, проставить размеры можно путем установки значений управляющих (системных) размерных переменных. С этой целью определим функцию (SETDIM...).

Управляющие размерные переменные (их около 40) начинаются с букв DIM... (DIMENSION – размер). Почти все они имеют значения по умолчанию и чтобы узнать их, достаточно набрать в командной строке AutoCAD команду DIM, а в строке запросов ключевое слово STATUS:

```
Command: DIM
```

```
Dim: STATUS
```

Появится текстовое окно AutoCAD с фрагментом списка размерных переменных и их текущими значениями (рис. 3.9). С помощью вертикальной

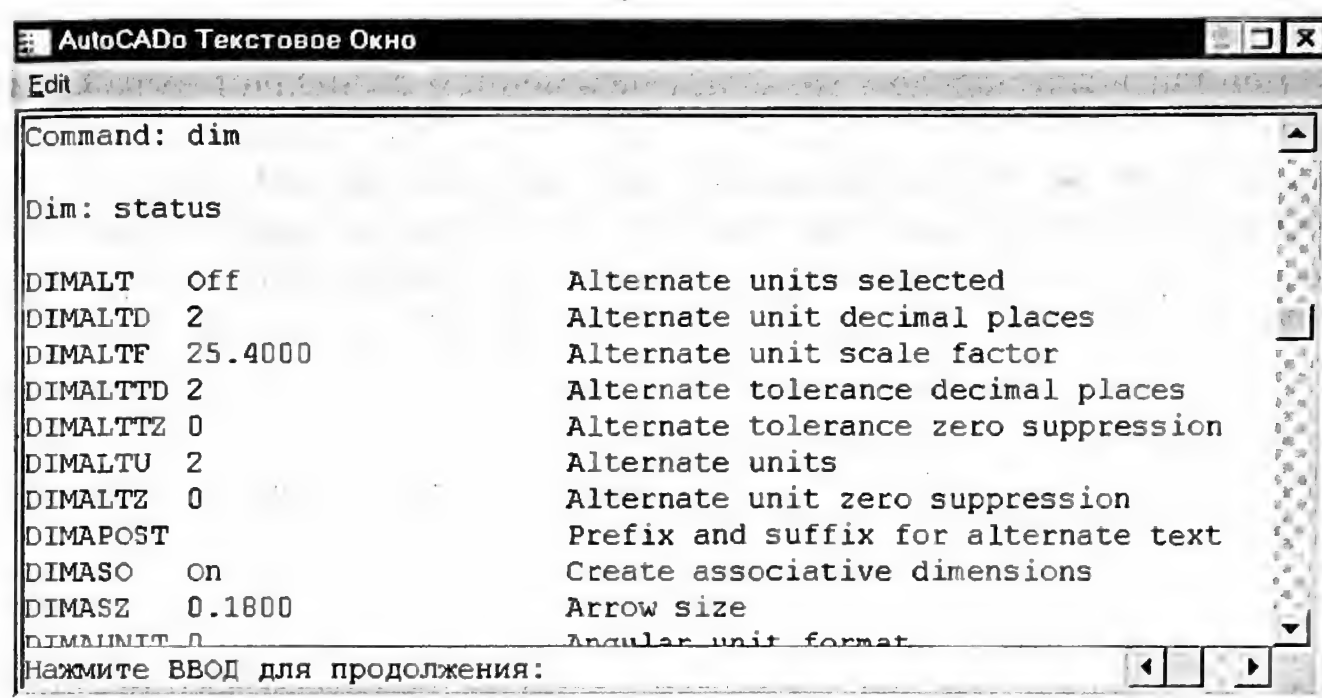


Рис. 3.9. Текстовое окно AutoCAD с фрагментом списка размерных переменных и их текущих значений

полосы прокрутки можно просмотреть все значения размерных системных переменных.

Состояние ON/OFF означает, что размерная переменная установлена/не установлена. В AutoLISP состоянию OFF соответствует значение 0, состоянию ON – 1. В нашей задаче функция настройки размерных переменных (**SETDIM...**) выглядит следующим образом:

```
(DEFUN SETDIM)
  (RMAX) ; функция настройки размерных переменных
  (COMMAND "UNITS" "2" "0" "1" "1" "0" "N") ; система единиц
  (SETVAR "DIMTAD" 1) ; текст над размерной линией
  (SETVAR "DIMSOXD" 1) ; текст между выносными линиями
  (SETVAR "DIMTIH" 0) ; текст, параллельный размерной линии
  (SETVAR "DIMDLI" (* RMAX 0.25)) ; отступ между линиями
  (SETVAR "DIMEXE" (* RMAX 0.05)) ; увеличение размерных линий
  (SETVAR "DIMTSZ" 0) ; изображение стрелки, а не засечки
  (SETVAR "DIMASZ" (* RMAX 0.15)) ; длина стрелки
)
```

RMAX – максимальный радиус из всех ступеней объекта.

Теперь перейдем к указанию размеров многоступенчатого объекта с помощью AutoLISP.

Формирование цикла для указания размера диаметра ступени объекта можно организовать с помощью функции (**FOREACH...**).

```
(FOREACH <переменная> <список> <выражение>... )
```

Эта функция последовательно присваивает значения элементов списка переменной и при каждом значении переменной вычисляет выражения, начиная с первого элемента списка. Процесс осуществляется до тех пор, пока не будут вычислены выражения для всех значений элементов списка.

Выделить необходимую пару точек начала выносных линий из списка пар точек можно с помощью параметров функции (**FOREACH...**). Для этого функция (**FOREACH...**) должна быть записана следующим образом:

```
(FOREACH EL LT2T3 <выражение>...)
```

где EL – переменная, которой присваивается значение элемента списка LT2T3. В этом списке содержатся пары точек, а выражение – это некоторая последовательность функций AutoLISP.

Выделить из пары точку начала первой выносной линии можно с помощью функции (**CAR <список>**), которая выделяет первый элемент списка.

Если, к примеру, мы рассматриваем трехступенчатый объект, то список LT2T3 будет иметь следующий вид: '('(T32 T33) '(T22 T23) '(T12 T13)).

В цикле переменная EL будет последовательно содержать список пары точек правого торца ступени объекта. Функция (**CAR...**) позволит выделить из списка пары точек точку начала первой выносной линии.

Выделение из пары точки начала второй выносной линии можно осуществить с помощью функции (**CADR...**).

Сохраним значения точек начала первой и второй выносных линий в переменных T2 и T3 соответственно с помощью функции (**SETQ...**):

```
(SETQ T2 (CAR EL)
      T3 (CADR EL))
```

Определить точки положения размерной линии (TR) можно с помощью следующего выражения:

```
(SETQ TR (LIST (- (CAR T2) 10) (CADR T2)))
```

Точка T2 – это список '(X2 Y2), где X2 – координата X точки T2; Y2 – координата Y точки T2.

Функция (**CAR T2**) выделяет значение координаты X точки T2. Из этого значения вычитается 10 единиц, в результате получается значение координаты X точки T2. Выражение (**CADR T2**) определяет значение координаты Y точки TR. С помощью функции (**LIST...**) образуется пара – список из двух координат, то есть точка TR, через которую должна пройти размерная линия.

Указание размера диаметра ступени объекта выполняется командами графического редактора AutoCAD с помощью функции (**COMMAND...**):

```
(COMMAND "DIM1" "VER" T2 T3 TR "%C<>")
```

"DIM1" – команда для указания размеров. Возможны несколько видов указания линейных размеров. Они отличаются только углом, под которым

проводится размерная линия. Для каждого вида размерной линии имеется свой ключ:

"HOR" – HORIZontal – горизонтальная размерная линия;

"VER" – VERtical – вертикальная размерная линия;

"ALI" – ALigned – линия, параллельная начальным точкам выносной линии;

"ROT" – ROTated – линия, повернутая на заданный угол.

После ключа **"VER"** задаются T2 и T3 – точки начала выносных линий, затем TR – точка положения размерной линии. Текстовая константа обеспечивает вывод знака диаметра перед значением точек T2 и T3.

Для ввода в размерный текст специальных символов, например, символа диаметра, используются управляющие коды:

%%o – включение/выключение режима надчеркивания;

%%u – включение/выключение режима подчеркивания;

%%d – специальный символ «градус» ($^{\circ}$);

%%p – специальный символ «допуска» (\pm);

%%c – специальный символ «диаметр» (\varnothing);

%%% – вывод одиночного символа «процент»;

%%nnn – специальный символ с десятичным кодом nnn.

Определить точку положения размерной линии при указании размеров длины ступеней объекта можно с помощью следующего выражения:

```
(SETQ TR (LIST X (- Y (* RMAX 1.2)))  
I 0)
```

Проставлять размеры ступеней объекта начнем с первой ступени, а список пар точек, который был создан ранее, содержит пары точек, начиная с последней ступени объекта. Это означает, что переменная LT2T3 содержит список **'(T32 T33) (T22 T23) (T12 T13)**). Для того чтобы изменить порядок элементов в списке на обратный, в AutoLISP имеется встроенная функция **(REVERSE...)**.

```
(REVERSE <список>)
```

После реверсирования переменная LT2T3 будет содержать список **'(T12 T13) (T22 T23) (T32 T33)**.

Первая пара точек соответствует первой ступени объекта, вторая – второй и т. д.

Выделить точку начала второй выносной линии для того, чтобы проставить размер длины ступени объекта, можно с помощью следующего выражения:

```
(SETQ T3 (CADR EL)  
I (+ I 1))
```

Проставить размер длины первой ступени объекта можно с помощью команды AutoCAD для указания размера:

```
(COMMAND "DIM" "HOR" BP T3 TR " ")
```


Здесь используется команда "DIM". Отличие команд "DIM1" и "DIM" заключается в том, что команда "DIM1" позволяет проставить один размер, а команда "DIM" — последовательность размеров.

Формирование цикла для указания размеров длин ступеней объекта можно осуществить с помощью функции (FOREACH...):

```
; цикл образмеривания длины ступени объекта
(FOREACH EL LT2T3
  (SETQ T3 (CADR EL) ; точка начала 2-й выносной линии;
    I (+ I 1))
  (IF (= I 1)
    (COMMAND "DIM" "HOR" BP T3 TR "") ; указание длины 1-й ступени
    (COMMAND "BAS" T3 "")) ; указание длины всех ступеней, кроме 1-й
)
```

Заметим, что размеры длины ступеней представляют собой последовательность базовых размеров.

Последовательность связанных размеров в графическом редакторе AutoCAD строится с помощью команд "BAS" — BASELINE (базовый) и "CON" — CONTINUE (продолженный).

Таким образом, чтобы проставить размеры длины всех ступеней, кроме 1-й, следует использовать команду BASELINE. Ниже представлена функция (MNOGOSTR...) создания и указания размеров многоступенчатого объекта.

3.3.4. Программа параметрического изображения и указания размеров многоступенчатого объекта

```
; *****
; (MNOGOSTR X Y LD) - функция создания изображения
; многоступенчатого объекта и указания его размеров
; А р г у м е н т ы  ф у н к ц и и:
; X, Y - координаты базовой точки объекта по осям X и Y
; LD - список диаметров и длин ступеней объекта вида
; '( (D1 L1) (D2 L2) (D3 L3) . . . )
; (SETDIM RMAX) - функция установки размерных переменных для
; определения вида размерных линий и самих размеров
; RMAX - максимальный радиус из всех ступеней объекта
; (PAZMV) - функция указания размеров (образмеривания) чертежа
; (STUP) - функция создания ступени объекта
; Пример вызова функции (MNOGOSTR X Y LD)
; (MNOGOSTR 10 100 ' ((40 70) (60 50) (40 30) (70 100)))
; *****
```

```

(DEFUN SETDIM (RMAX) ; функция определения вида размерных линий
  (COMMAND "UNITS" "2" "0" "1" "1" "0" "N") ; система единиц
  (SETVAR "DIMITAD" 1) ; текст над размерной линией
  (SETVAR "DIMSOXD" 1) ; текст между выносными линиями
  (SETVAR "DIMITH" 0) ; текст расположен параллельно линии
  (SETVAR "DIMDLI" (* RMAX 0.25)) ; отступ между линиями
  (SETVAR "DIMEXE" (* RMAX 0.05)) ; увеличение размерных линий
  (SETVAR "DIMTSZ" 0) ; изображение стрелки на
  ; концах размерной линии
  (SETVAR "DIMASZ" (* RMAX 0.15)) ; длина стрелки
)

(DEFUN RAZMV ()
  (SETDIM RMAX) ; вызов функции определения вида размерных линий
  (SETQ LT2T3 (REVERSE LT2T3))
  (COMMAND "STYLE" "" "COMPLEX" (* RMAX 0.15) "1" "" "" "" "")
  (FOREACH EL LT2T3 ; цикл указания размеров диаметров ступеней
    (SETQ T2 (CAR EL) ; точка для проведения 1-й выносной линии
      T3 (CADR EL) ; точка для проведения 2-й выносной линии
      TR (LIST (- (CAR T2) 10) (CADR T2)))
    (COMMAND "DIM1" "VER" T2 T3 TR "%%c <>")
  )
  (SETQ TR (LIST X (- Y (* RMAX 1.2)))
    I 0)
  (FOREACH EL LT2T3 ; цикл указания длины ступеней
    (SETQ T3 (CADR EL) ; точка проведения 2-й выносной линии
      I (+ I 1))
    (IF (= I 1)
      (COMMAND "DIM" "HOR" BP T3 TR "") ; изображение размера
      (COMMAND "BAS" T3 ""))) ; 1-й ступени и последующих
  )

(DEFUN STUP ()
  (SETQ T10 (POLAR BT (/ PI 2) R0)
    T1 (POLAR BT (/ PI 2) R)
    T2 (POLAR T1 0 L)
    T3 (POLAR T2 (* 1.5 PI) D)
    T4 (POLAR T3 PI L)
    T40 (POLAR BT (* 1.5 PI) R0)
    BT (POLAR BT 0 L))
  (COMMAND "PLINE" T10 "W" 0.2 0.2 T1 T2 T3 T4 T40 "")
)

(DEFUN MNOGOSTR (X Y LD) ; начало создания функции
  (TEXTSCR) ; переход на текстовый экран
  (SETVAR "CMDECHO" 0)
  (SETVAR "BLIPMODE" 0)
  (COMMAND "LIMITS" "0,0" "200,120" "ZOOM" "A")

```

```

(SETQ RMAX 0 LT2T3 '())
  R0 0 BT (LIST X Y)
  BP BT)
(Foreach EL LD                                ; начало цикла создания ступени
  (SETQ D (CAR EL)                             ; выделение из пары диаметра ступени
    L (CADR EL)                               ; выделение из пары длины ступени
    R (* D 0.5))
  (IF (> R RMAX) (SETQ RMAX R)) ; определение максимального
                                ; радиуса
  (IF (< R R0) (SETQ R0 R))
  (STUP)                                       ; вызов функции для создания
                                ; ступеней
  (SETQ LT2T3 (CONS (LIST T2 T3) LT2T3)      ; список пар точек
    R0 R))                                   ; для изображения выносных линий
(RAZMV)                                       ; вызов функции для указания
                                ; размеров чертежа
(COMMAND "LAYER" "N" "OSI" "L" "CENTER" "OSI" "S" "OSI"
"" "LINE" BP BT "")
(PRIN1)
)
(MNOGOSTR 10 60 '((30 40) (40 50) (30 30) (15 40))) ; вызов функции

```

Результатом выполнения функции (MNOGOSTR...) будет изображение многоступенчатого объекта с указанием его размеров, аналогичное изображенному на рис. 3.8.

Чтобы завершить оформление чертежа, необходимо вывести текстовые пояснения, которые сначала можно поместить в текстовый файл, а затем на чертеж.

Приведем функцию (TEXTIN1), которая считывает информацию из текстового файла и помещает ее на чертеж в указанную точку. Обращение к текстовому файлу, местоположение информации производится в диалоговом режиме.

3.3.5. Программа параметрического ввода текста из файла на чертеж

```

; *****
; (TEXTIN1) - функция размещения текста из файла на чертеже
; *****
(DEFUN TEXTIN1 (/ FN BP OF RL P U)
  (SETVAR "CMDECHO" 0) ; отключение эха команд
  (SETVAR "BLIPMODE" 0) ; отключение изображения маркера
  (COMMAND "LIMITS" "0,0" "297,210" "ZOOM" "A")
  (SETQ FN (GETSTRING "\n Введите имя файла с текстом : "))

```

```

BP (GETPOINT "\n Введите точку ввода текста: [можно мышкой] ")
P (GETREAL "\n Введите высоту букв текста: ")
U (GETREAL "\n Введите угол наклона текста: "))
(TYPE FN)
(PRINT FN)
(SETQ OF (OPEN FN "r")
  H (/ P 4)
  RL (READ-LINE OF))
(COMMAND "STYLE" "" "" P H "" "" "" ""
  "TEXT" BP U RL)
(WHILE (SETQ RL (READ-LINE OF))
  (COMMAND "TEXT" "" RL))
(CLOSE OF)
)
(TEXTIN1) ; вызов функции (TEXTIN1)

```

Функция (**TEXTIN1**) обеспечивает считывание заданного текстового файла построчно. Полное имя файла запоминается в локальной переменной FN (File Name – имя файла). В локальной переменной BP (Base Point – базовая точка) запоминается начальная точка первой строки (записи) текста, в локальной переменной P – высота букв текста, в переменной U – угол наклона текста. В локальной переменной OF запоминается полное имя файла, открытого для чтения "r" (read). В локальной переменной RL запоминается считываемая строка текста.

Функция (**COMMAND...**) выводит на чертеж первую строку, взятую из текстового файла. Для вывода всех записей в текстовом файле используется функция (**WHILE...**).

Выражение

```
(SETQ RL (READ-LINE OF))
```

обеспечивает считывание очередной записи из открытого файла (TEXT.LSP) и ее сохранение в переменной RL для последующего вывода на чертеж с помощью команды TEXT.

Для вывода на чертеж текстовых данных (например, технических требований из заранее подготовленного текстового файла) функцию (**TEXTIN1**) можно включить в функцию (**MNOGOSTD**) или (**MNOGSTPM**). Для этого необходимо в функции (**MNOGOSTD**) или (**MNOGSTPM**) перед последней круглой скобкой разместить функцию (**TEXTIN1**) с указанием полного имени, например:

```
(LOAD "C:/ALISP/TEXTIN1")
(TEXTIN1)
```

Допустим, сверху в поле чертежа многоступенчатого объекта (см. рис. 3.7) вам предстоит ввести надпись:

ГРАФИЧЕСКОЕ ИЗОБРАЖЕНИЕ МНОГОСТУПЕНЧАТОГО ОБЪЕКТА

Прежде всего необходимо создать текстовый файл с этой записью, например, файл TEXT.TXT в каталоге ALISP на диске C:. Пусть в этом же каталоге находятся файлы MNOGOSTPM.LSP и TEXTIN1.LSP. Затем в командной строке AutoCAD необходимо загрузить файл MNOGOSTPM.LSP, в нем находится функция (MNOGOSTPM...), которую следует выполнить.

```
Command: (LOAD "C:/ALISP/MNOGOSTPM")
```

```
MNOGOSTPM
```

```
Command: (MNOGOSTPM 10 60 ' ((30 40) (40 50) (30 30) (15 40)))
```

На экране появится графическое изображение многоступенчатого объекта (см. рис. 3.7). Затем предстоит загрузить файл TEXTIN1.LSP, в нем находится функция (TEXTIN1), которую необходимо выполнить.

```
Command: (LOAD "C:/ALISP/TEXTIN1")
```

```
TEXTIN1
```

```
Command: (TEXTIN1)
```

В окне появятся следующие запросы:

Введите имя файла с текстом : C:/ALISP/TEXT.TXT

Введите точку ввода текста: [можно мышкой] 15,100

Введите высоту букв текста 4

Введите угол наклона текста 0

После этого в поле чертежа появится первая строчка из файла TEXT.TXT, затем вторая. Начало текста находится в точке с координатами (15,100), с высотой букв 4 и наклоном текста 0 градусов.

3.4. Создание параметрических изображений проекций конструкций

Основные этапы автоматизации создания параметрического изображения проекций конструкции рассмотрим на примере ферм.

3.4.1. Постановка задачи

Требуется разработать алгоритм и программу для автоматизации создания параметрического изображения фронтальной проекции фермы с заданным числом секций – N, заданной длиной фермы – L и заданной высотой – H. Общий вид фронтальной проекции фермы представлен на рис. 3.10. Необходимо предусмотреть возможность изменения всех параметров фермы: длины, высоты, числа секций, а также точки подвеса фермы. Следует разработать два варианта программы в диалоговом и пакетном режимах. В первом случае необходимая исходная информация запрашивается в процессе выполнения программы; во втором – вся исходная информация задается до выполнения программы в виде аргументов функции.

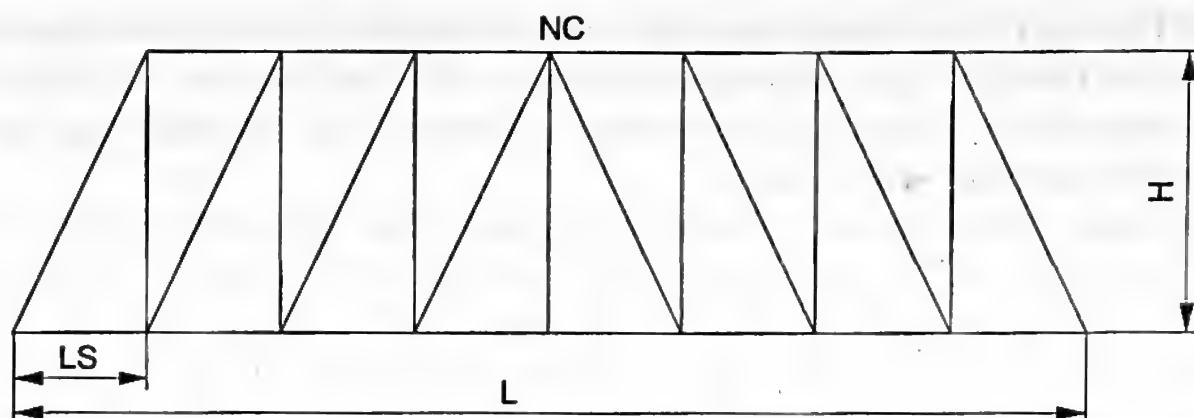


Рис. 3.10. Общий вид фронтальной проекции фермы

3.4.2. Выявление основных особенностей, взаимосвязей и количественных закономерностей

В ходе анализа общего вида проекции фермы (см. рис. 3.10) можно заметить, что конструкция прямоугольной секции до точки подвеса фермы – NC отличается от конструкции прямоугольной секции, расположенной после точки подвеса, наклоном раскоса. Это обстоятельство можно использовать для упрощения последовательности действий и программы.

Конструкция фермы включает две угловые и остальные прямоугольные секции. Для создания прямоугольных секций необходимо знать номер секции подвеса фермы – NC, длину секции – LS, которая в данном случае равна L/N . Число прямоугольных секций (NS) на две меньше общего числа, то есть $NS = N - 2$. Создание параметрического изображения проекции фермы необходимо начинать с базовой точки (точки начала прорисовки фермы – ВТФ), задавая ее координаты X и Y. Будем считать, что все секции фермы имеют одинаковую длину.

3.4.3. Разработка последовательности действий и программы параметрического изображения фронтальной проекции фермы

Создание параметрического изображения фронтальной проекции фермы состоит из нескольких основных этапов.

Создание начальной угловой секции. При диалоговом режиме работы необходимо предусмотреть ввод следующих данных:

- XF – координата x базовой точки проекции фермы;
- YF – координата y базовой точки проекции фермы;
- L, H – длина и ширина фермы соответственно;
- N, NC – число секций в ферме и номер секции подвеса фермы соответственно.

Для создания изображения начальной угловой секции требуется знание трех точек (см. рис. 3.11, а): базовой точки – ВТФ; верхней – Т1 и нижней – Т2. Стрелками на рис. 3.11 показаны схемы создания параметрического изображения угловой секции.

Создание прямоугольных секций фермы. Для создания графического изображения каждой прямоугольной секции достаточно знать координаты четырех точек: Т1, Т2, Т3, Т4. При этом для создания графического изображения каждой последующей прямоугольной секции достаточно определить координаты только двух точек – Т3 и Т4. Точки Т1 и Т2 в новой строящейся секции совпадают с точками Т3 и Т4, определенными для предыдущей секции. Схемы создания графического изображения прямоугольных секций показаны на рис. 3.11, б, в.

Создание графического изображения секций следует проводить в цикле. Если секция расположена до точки подвеса фермы, прорисовка секции проводится по схеме рис. 3.11, б; если прямоугольная секция находится после точки подвески, то она изображается по схеме рис. 3.11, в.

Создание конечной угловой секции. Для создания конечной угловой секции достаточно знать координаты только Т3 – точки конца фермы; координаты еще двух точек (Т1 и Т2) можно взять из предыдущего цикла.

Далее рассмотрим функции AutoLISP, которые следует использовать для создания графического изображения фронтальной проекции фермы.

Ввод исходных данных в диалоговом режиме можно представить в таком виде:

```
(SETQ XF (GETREAL "\n Координата x базовой точки фермы XF = ")
      YF (GETREAL "\n Координата y базовой точки фермы YF = ")
      L (GETREAL "\n Длина фермы L = ")
      NC (GETINT "\n Номер секции подвеса фермы NC = ")
      H (GETREAL "\n Высота фермы H = ")
      N (GETINT "\n Число секций в ферме N = ")
```

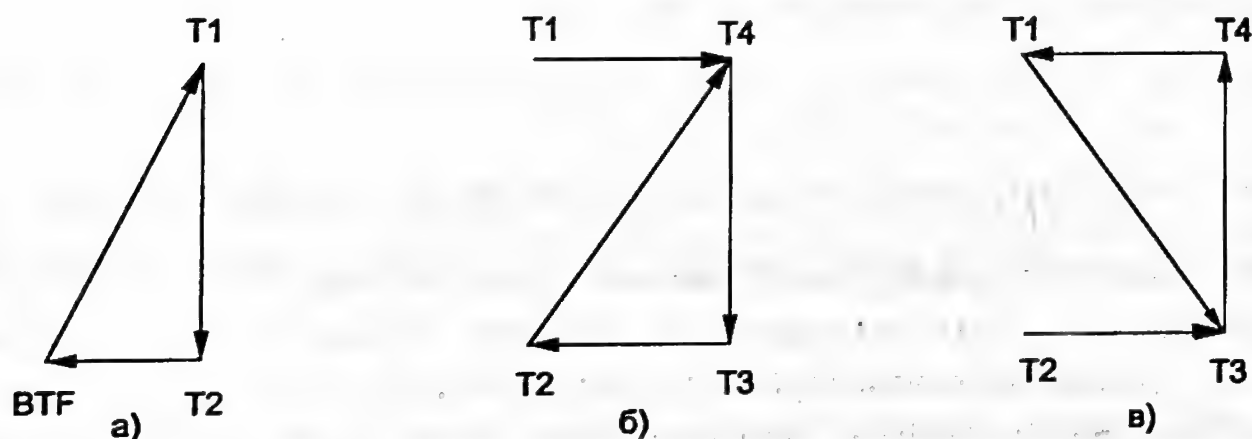


Рис. 3.11. Угловые и прямоугольные секции фермы

Для создания начальной угловой секции определим базовую точку BTF, длину секции LS, точки T1, T2 и используем команду "PLINE";

```
(SETQ BTF (LIST XF YF)
  LS (/ L N)
  T2 (POLAR BTF 0 LS)
  T1 (POLAR T2 (/ PI 2) H)
  I 0
  NS (- N 2))
)
(COMMAND "PLINE" BTF "W" 0.2 0.2 T1 T2 "C")
```

Создание прямоугольных секций выполним в цикле с использованием функции (REPEAT...). При создании каждой следующей прямоугольной секции необходимо проверить условие перехода через точку подвеса, так как при этом изменяется вид секции. В конце цикла переприсвоим координаты последних двух точек последней изображенной секции двум точкам T1 и T2, которые служат начальными точками для создания следующей прямоугольной секции.

Эта часть программы будет выглядеть так:

```
(REPEAT NS
  (SETQ T3 (POLAR T2 0 LS)
    T4 (POLAR T1 0 LS)
    I (+ I 1))
  (IF (< I NC)
    (COMMAND "PLINE" T1 "W" 0.2 0.2 T4 T3 T2 T4 "")
    (COMMAND "PLINE" T2 "W" 0.2 0.2 T3 T4 T1 T3 ""))
  (SETQ T2 T3 T1 T4)
)
```

Создание конечной угловой секции можно выполнить следующим образом:

```
(SETQ TN (POLAR T2 0 LS))
(COMMAND "PLINE" T4 "W" 0.2 0.2 TN T3 "")
```

Ниже представлена функция для создания параметрического изображения фронтальной проекции фермы (SFD...).

3.4.4. Программа параметрического изображения фронтальной проекции фермы в диалоговом режиме

```
; *****
; (SFD) - функция создания параметрического изображения
; фронтальной проекции фермы в диалоговом режиме
; *****
```



```

(DEFUN SFD ()
  (SETVAR "CMDECHO" 0) ; отключение эха команд редактора
  (SETVAR "BLIPMODE" 0) ; отключение изображения маркера
  (COMMAND "LIMITS" "0,0" "297,210" "ZOOM" "A")
  (TEXTSCR) ; переход на текстовый экран
  (PROMPT "\n Введите основные параметры фермы:")
  (SETQ
    XF (GETREAL "\n Координата x базовой точки фермы XF = ")
    YF (GETREAL "\n Координата y базовой точки фермы YF = ")
    L (GETREAL "\n Длина фермы L = ")
    NC (GETINT "\n Номер секции подвеса фермы NC = ")
    H (GETREAL "\n Высота фермы H = ")
    N (GETINT "\n Число секций в ферме N = "))
  (SETQ BT (LIST XF YF) ; определение базовой точки BT
    LS (/ L N) ; определение длины секции LS
    T2 (POLAR BT 0 LS)
    T1 (POLAR T2 (/ PI 2) H)
    I 0
    NS (- N 2))
  (COMMAND "PLINE" BT "W" 0.2 0.2 T1 T2 "C")
  (REPEAT NS ; определение точек секции и создание
    (SETQ T3 (POLAR T2 0 LS) ; изображения фермы
      T4 (POLAR T1 0 LS)
      I (+ I 1))
    (IF (< I NC)
      (COMMAND "PLINE" T1 "W" 0.2 0.2 T4 T3 T2 T4 "")
      (COMMAND "PLINE" T2 "W" 0.2 0.2 T3 T4 T1 T3 ""))
    (SETQ T2 T3 T1 T4))
  (SETQ TN (POLAR T2 0 LS))
  (COMMAND "PLINE" T4 "W" 0.2 0.2 TN T3 ""))
(SFD) ; вызов функции

```

Результат выполнения функции (SFD)

Введите основные параметры фермы:

Координата x базовой точки фермы XF = 10

Координата y базовой точки фермы YF = 130

Длина фермы L = 160

Номер секции подвеса фермы NC = 6

Высота фермы H = 42.0

Число секций в ферме N = 8

Результатом выполнения функции (SFD) будет параметрическое изображение фермы (рис. 3.12).

Для создания функции, обеспечивающей формирование параметрического изображения аналогичной фермы, но в пакетном режиме, достаточно создать функцию с параметрами (SF XF YF L H B N NC UF).

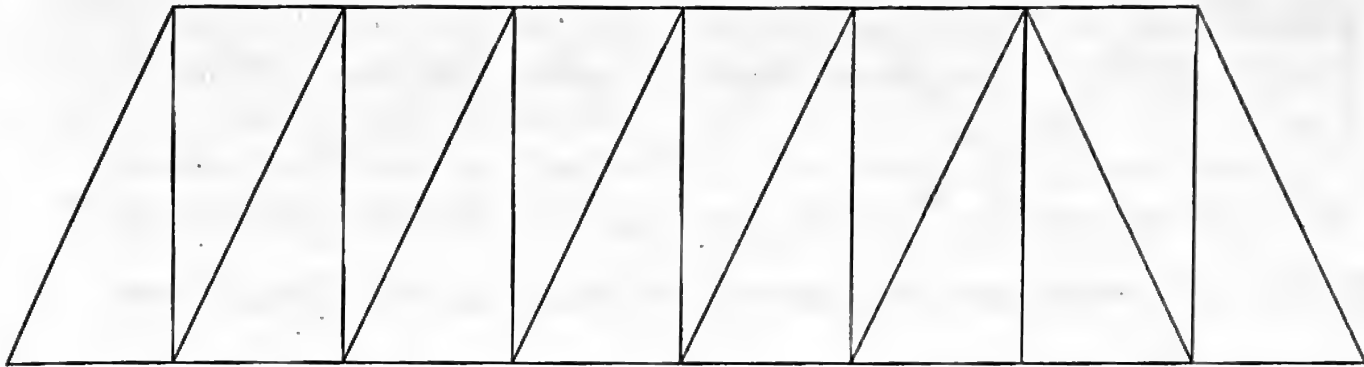


Рис. 3.12. Параметрическое изображение фермы,
созданное при помощи функции (SFD)

3.4.5. Программа параметрического изображения горизонтальной проекции фермы

```
; *****
; (SG XF YF L B N) - функция создания параметрического
; изображения горизонтальной проекции фермы
; А р г у м е н т ы   ф у н к ц и и:
; XF,YF - координаты базовой точки фермы по оси X,Y
; L,B - длина, ширина фермы
; N - число секций в ферме (четное число секций/
; *****
(DEFUN SG (XF YF L B N)           ; начало создания функции
  (SETVAR "CMDECHO" 0)           ; отключение эха команд
  (SETVAR "BLIPMODE" 0)          ; отключение изображения маркера
  (SETQ BTG (LIST XF YF))        ; определение базовой точки
  LS (/ L N)                      ; определение длины секции
  T2 BTG
  T1 (POLAR T2 (/ PI 2) B))
(COMMAND "PLINE" T2 "W" 0.2 0.2 T1 "")
(REPEAT (/ N 2)
  (SETQ T3 (POLAR T2 0 LS)
    T4 (POLAR T1 0 LS))
  (COMMAND "PLINE" T1 "W" 0.2 0.2 T4 T3 T2 T4 "")
  (SETQ T2 T3 T1 T4)
  (SETQ T3 (POLAR T2 0 LS)
    T4 (POLAR T1 0 LS))
  (COMMAND "PLINE" T2 "W" 0.2 0.2 T3 T4 T1 T3 "")
  (SETQ T2 T3 T1 T4))
)
(SG 40 200 320 40.0 8)           ; вызов функции
```

Результатом выполнения функции (SG...) будет параметрическое изображение фермы (рис. 3.13).

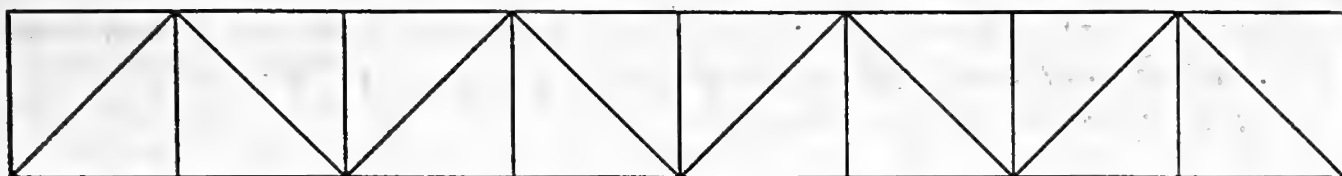


Рис. 3.13. Параметрическое изображение фермы, созданное при помощи функции (SG...)

3.4.6. Программа параметрического изображения изометрии фермы

Ниже приведена программа создания изометрии фермы.

```
; *****
; (SF XF YF L H N NC UF) - функция изображения фронтальной
; проекции фермы
; А р г у м е н т ы   ф у н к ц и и:
; XF- координата x базовой точки фермы
; YF- координата y базовой точки фермы
; L - длина фермы
; H - высота фермы
; B - ширина фермы
; N - число секций в ферме
; NC- номер секции подвеса фермы
; UF- угол наклона фронтальной проекции
; *****
(DEFUN SF (XF YF L H B N NC UF)
  (SETQ BTF (LIST XF YF)
    LS (/ L N)
    T2 (POLAR BTF 0 LS)
    T1 (POLAR T2 UF H)
    TV T1
    I 0
    NS (- N 2))
  (COMMAND "PLINE" T2 "W" 0.2 0.2 BTF T1 "C")
  (REPEAT NS
    (SETQ T3 (POLAR T2 0 LS)
      T4 (POLAR T1 0 LS)
      I (+ I 1))
    (IF (< I NC)
      (COMMAND "PLINE" T1 "W" 0.2 0.2 T4 T3 T2 T4 "")
      (COMMAND "PLINE" T2 "W" 0.2 0.2 T3 T4 T1 T3 ""))
    )
  (SETQ T2 T3 T1 T4)
  )
  (SETQ TN (POLAR T2 0 LS))
  (COMMAND "PLINE" T4 "W" 0.2 0.2 TN T3 "")
  )
)
```

```

; *****
; SG-- функция создания параметрического изображения
; нижнего пояса фермы с наклоном
; B - ширина нижнего пояса фермы
; UG - угол наклона нижнего пояса
; *****
(DEFUN SG (XF YF L H B N NC UG)
  (SETQ BTG (LIST XF YF) LS (/ L N) T2 BTG
    T1 (POLAR T2 UG B) TN T1)
  (COMMAND "PLINE" T2 "W" 0.2 0.2 T1 "")
  (REPEAT (/ N 2)
    (SETQ T3 (POLAR T2 0 LS)
      T4 (POLAR T1 0 LS)
    )
    (COMMAND "PLINE" T1 "W" 0.2 0.2 T4 T3 T2 T4 "")
    (SETQ T2 T3 T1 T4)
    (SETQ T3 (POLAR T2 0 LS)
      T4 (POLAR T1 0 LS))
    (COMMAND "PLINE" T2 "W" 0.2 0.2 T3 T4 T1 T3 "")
    (SETQ T2 T3 T1 T4))
  )
; *****
; SI - функция создания параметрического изображения
; изометрии фермы
; *****
(DEFUN SI (XF YF L H B N NC UF UG)
  (SETVAR "CMDECHO" 0)
  (SETVAR "BLIPMODE" 0)
  (COMMAND "LIMITS" "0,0" "400,300" "ZOOM" "A")
  (SF XF YF L H B N NC UF) ; вызов функции
  (SG XF YF L H B N NC UG) ; вызов функции
  (REPEAT (- N 1)
    (SETQ TV1 (POLAR TV 0 LS)
      TN1 (POLAR TN 0 LS)
    )
    (COMMAND "PLINE" TN "W" 0.2 0.2 TV TN1 "")
    (SETQ TV TV1
      TN TN1)
  )
  (SETQ TN (POLAR TN 0 LS)
    TV (POLAR TV PI LS)
  )
  (COMMAND "PLINE" TV "W" 0.2 0.2 TN "")
)
(SI 40 200 320 62.0 40 8 4 1.2 0.5) ; вызов функции

```


Результатом выполнения функции (SI...) будет параметрическое изображение изометрии фермы (рис. 3.14).

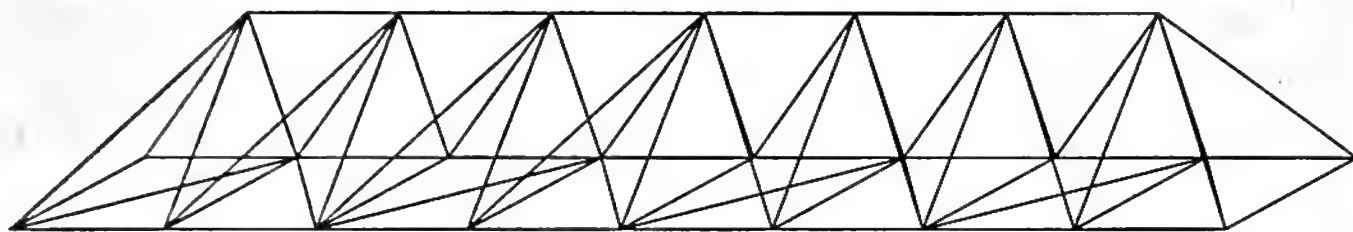


Рис. 3.14. Параметрическое изображение фермы, созданное при помощи функции (SI...)

3.5. Создание параметрического изображения общего вида объекта

Многие объекты отличаются друг от друга размерами тех или иных элементов, узлов и т. д., которые можно описать параметрически.

3.5.1. Постановка задачи

Требуется разработать алгоритм параметрического трехмерного изображения общего вида коттеджа с прорисовкой окон, дверей и даже форточек и создать функцию для параметрического изображения коттеджа (KOTTEDJ SO KR UXY UX), в которой:

SO – список координат точек и размеров элементов коттеджа вида:

((X1 Y1 Z1 L1 B1 H1) (X2 Y2 Z2 L2 B2 H2)...), где

X1, Y1, Z1 – координаты базовой точки коттеджа;

L1, B1, H1 – размеры коттеджа (длина, ширина, высота);

X_, Y_, Z_ – координаты базовых точек элементов коттеджа (окон, дверей, форточек);

L_ B_ H_ – размеры элементов коттеджа (окон, дверей, форточек): их длина, ширина и высота соответственно;

KR – высота крыши;

UXY, UX – углы направления взгляда на коттедж относительно плоскости XY и оси X.

3.5.2. Выявление основных особенностей, взаимосвязей и количественных закономерностей

Сделаем эскизную прорисовку общего вида коттеджа (рис. 3.15).

– Для параметрического изображения коттеджа используем трехмерные элементарные команды AutoCAD – параллелепипед "BOX" и клин

"WEDGE". Параллелепипед "BOX" нам потребуется для изображения самого коттеджа, окон, дверей и форточек, а клин "WEDGE" – для изображения крыши коттеджа.

В команде построения параллелепипеда "BOX" присутствуют следующие аргументы: базовая точка – BT; признак ввода размеров параллелепипеда – "L"; длина, ширина и высота параллелепипеда – соответственно LO, BO, HO.

3.5.3. Разработка последовательности действий и программы параметрического изображения коттеджа

Полное описание коттеджа зададим в виде списка SO. Он включает все элементы коттеджа в виде списка EL вида '(XO YO ZO LO BO HO). Чтобы использовать информацию для создания графического изображения элемента, ее необходимо предварительно извлечь из списка EL:

```
(SETQ XO (NTH 0 EL) ; координата x базовой точки элемента
      YO (NTH 1 EL) ; координата y базовой точки элемента
      ZO (NTH 2 EL) ; координата z базовой точки элемента
      LO (NTH 3 EL) ; длина элемента
      BO (NTH 4 EL) ; ширина элемента
      HO (NTH 5 EL) ; высота элемента
      I (+ I 1))
```

```
(COMMAND "BOX" (LIST XO YO ZO) "L" LO BO HO)
```

Для создания графического изображения крыши используем команду "WEDGE" со следующими аргументами: базовая точка – BT; признак ввода размеров клина – "L"; длина, ширина и высота клина – соответственно LO, BO, KR. Для определения базовой точки крыши необходимо использовать координаты базовой точки коттеджа и его размеры:

```
(IF (= I 1)
  (PROGN
    (SETQ LO2 (/ LO 2))
    (COMMAND
      "WEDGE" (LIST (+ XO LO2) YO HO) "L" (- 0 LO2) BO KR
      "WEDGE" (LIST (+ XO LO2) YO HO) "L" LO2 BO KR)))
```

В трехмерном пространстве выберем «точку зрения», то есть местоположение глаза наблюдателя относительно просматриваемого объекта (в нашем случае коттеджа). AutoCAD позволяет взглянуть на объект из любой точки пространства (даже изнутри изображаемого объекта). Линию от «точки зрения» до объекта часто называют линией взгляда или направлением взгляда.

Получить динамические трехмерные и перспективные виды можно с помощью команды **"DVIEW"**, в которой используется аналогия с камерой, направленной в сторону объекта. В качестве аргументов могут выступать: точка объекта, которая выбрана для просмотра, – ТО; признак просмотра камерой – "CA"; углы между направлением взгляда и плоскостью XY и угол между линией взгляда и осью X – соответственно UXY UX:

```
(COMMAND "DVIEW" (LIST XO YO) "" "CA" UXY UX "")
```

Угол UX изменяется в пределах от 180° до -180° – значение со знаком минус обозначает отсчет угла по часовой стрелке, со знаком плюс – против часовой стрелки.

Угол UXY изменяется в пределах от 90° до -90° – значение со знаком минус обозначает отсчет угла вниз от плоскости XY, со знаком плюс – вверх.

Используем направление взгляда вдоль линии, лежащей под углом $UXY = 40^\circ$ к плоскости XY, проекция которой на плоскость XY находится под углом $UX = -140^\circ$ относительно оси X.

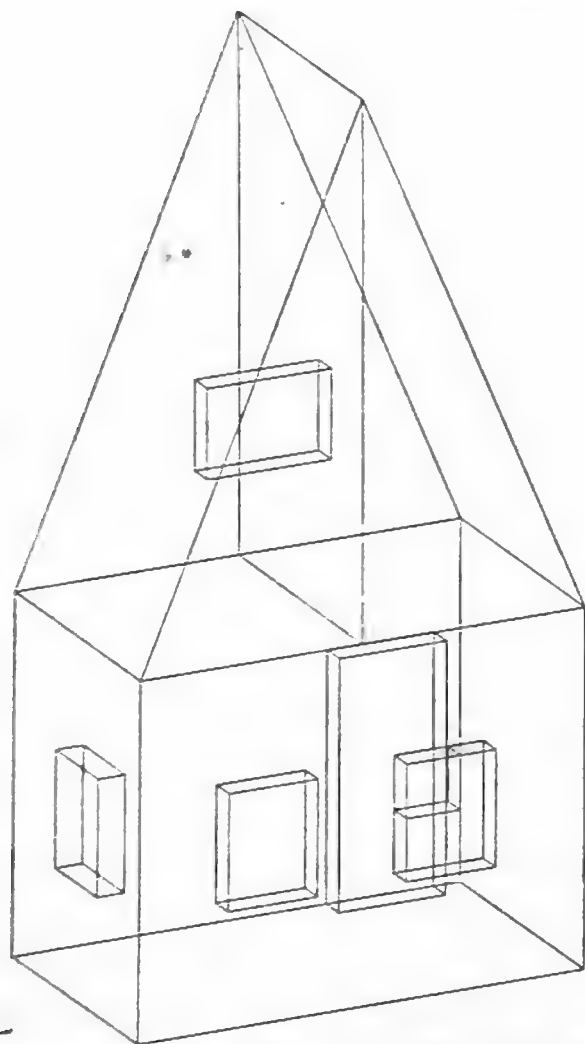


Рис. 3.15. Эскиз общего вида коттеджа

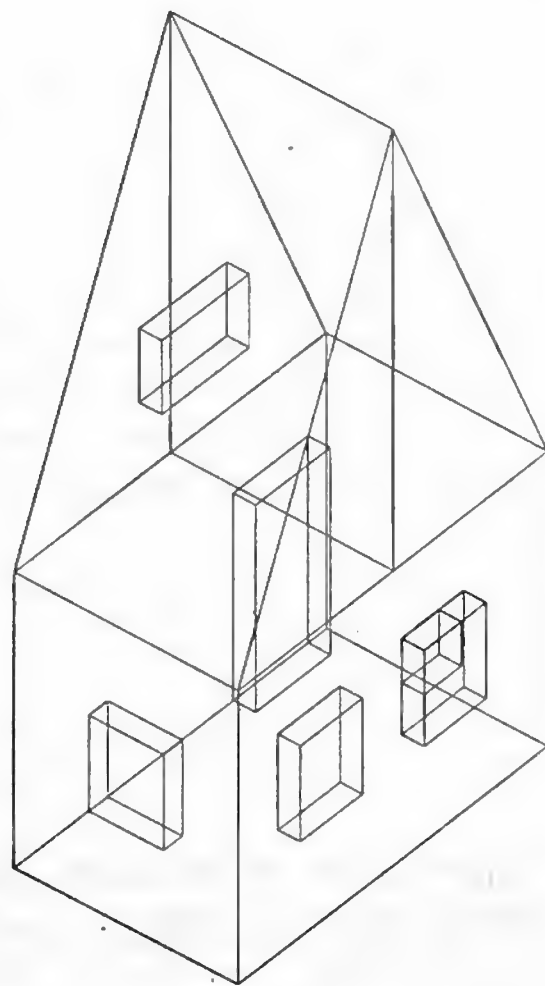


Рис. 3.16. Параметрическое изображение коттеджа, созданное при помощи функции (KOTTEDJ...)

3.5.4. Программа параметрического изображения коттеджа

```
; *****
; (KOTTEDJ SO KR UXY UX) - функция создания
; параметрического изображения коттеджа
; А р г у м е н т ы   ф у н к ц и и:
; SO - список базовых точек и размеров элементов коттеджа
; (LIST (LIST X1 Y1 Z1 L1 B1 H1) (LIST X2 Y2 Z2 L2 B2 H2)...)
; X1, Y1, Z1 - координаты базовой точки коттеджа
; L1, B1, H1 - размеры коттеджа (длина, ширина, высота)
; X_, Y_, Z_ - координаты базовых точек элементов коттеджа
; L_ B_ H_ - размеры элементов коттеджа (длина, ширина, высота)
; KR - высота крыши
; UXY, UX - углы взгляда на коттедж с плоскостью XY и с осью X
; Пример вызова функции (KOTTEDJ SO KR UXY UX)
; (KOTTEDJ (LIST (LIST 20 20 0 50 30 40) (LIST 30 20 13 10 3 13)
; (LIST 50 20 13 10 3 13) (LIST 55 50 0 12 -3 28)) 60 40 -140)
; *****
(DEFUN KOTTEDJ (SO KR UXY UX)
  (COMMAND "LIMITS" "0,0" "300,400" "ZOOM" "A")
  (SETQ I 0)
  (FOREACH EL SO
    (SETQ XO (NTH 0 EL)
      YO (NTH 1 EL)
      ZO (NTH 2 EL)
      LO (NTH 3 EL)
      BO (NTH 4 EL)
      HO (NTH 5 EL)
      I (+ I 1))
    (COMMAND "BOX" (LIST XO YO ZO) "L" LO BO HO)
    (IF (= I 1) (PROGN (SETQ LO2 (/ LO 2))
      (COMMAND
        "WEDGE" (LIST (+ XO LO2) YO HO) "L" (- 0 LO2) BO KR
        "WEDGE" (LIST (+ XO LO2) YO HO) "L" LO2 BO KR
        "DVIEW" (LIST XO YO) "" "CA" UXY UX ""))
      )
    )
  )
)
(KOTTEDJ (LIST (LIST 20 20 0 50 30 40)
  (LIST 30 20 13 10 3 13)
  (LIST 50 20 13 10 3 13)
```



```
(LIST 20 30 13 3 10 13)  
(LIST 55 50 0 12 -3 28)  
(LIST 40 50 50 14 -3 10)  
(LIST 50 20 20 6 3 6)) 60 40 -140)
```

Результатом выполнения функции (KOTTEDJ...) будет параметрическое изображение коттеджа (рис. 3.16).

3.6. Создание параметрического изображения общего вида машины

3.6.1. Постановка задачи

Требуется разработать функции на языке AutoLISP для формирования параметрического изображения основных узлов экскаватора и всего экскаватора в целом (как в диалоговом, так и пакетном режимах), допуская изменение основных параметров узлов.

3.6.2. Выявление основных особенностей, взаимосвязей и количественных закономерностей

Выделим основные узлы экскаватора: ходовое устройство (ходовая часть, поворотная платформа); рабочее оборудование. На поворотной платформе должен быть оборудован привод – силовая установка, кабина и рабочее оборудование. В рабочем оборудовании необходимо выделить стрелу, рукоять и ковш, которые могут иметь различное конструктивное исполнение. Ограничимся рассмотрением гидравлического одноковшового экскаватора. Предусмотреть возможность прорисовки гидроцилиндров для наклона стрелы, рукояти и поворота ковша, а также возможность различного положения рабочего оборудования экскаватора.

3.6.3. Разработка последовательности действий и программы параметрического изображения основных узлов машины

Создание параметрического изображения общего вида машины включает следующие этапы:

- параметрическое представление основных узлов машины;
- выделение для каждого узла базовых точек (для самого узла, а также для узлов, связанных с ним);

- введение возможности конструктивного изменения отдельных узлов за счет изменения того или иного параметра, например, угла перелома в стреле экскаватора.

Начнем разработку функции с ходовой части экскаватора (рис. 3.17).

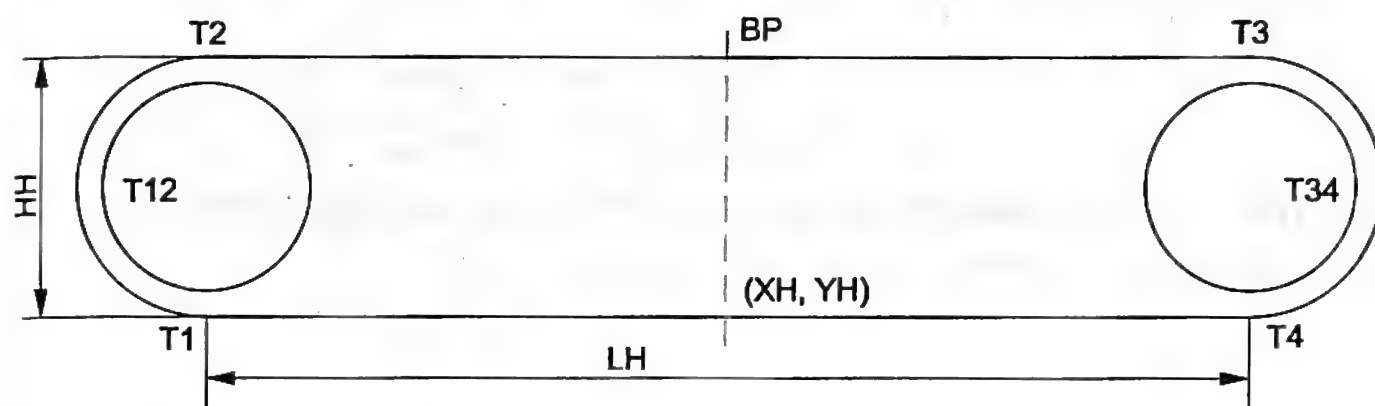


Рис. 3.17. Параметрическое представление ходовой части

3.6.4. Программа параметрического изображения ходовой части экскаватора

```
; *****
; (GUSHOD XH YH LH HH XP YP) - функция создания
; параметрического изображения гусеничного хода
; А р г у м е н т ы   ф у н к ц и и:
; XH - координата x базовой точки гусеничного хода
; YH - координата y базовой точки гусеничного хода
; LH - размер базы гусеничного хода
; HH - высота гусеничного хода
; XP - координата x базовой точки поворотной платформы
; YP - координата y базовой точки поворотной платформы
; *****
(DEFUN GUSH (XH YH LH HH XP YP)
  (SETQ BT (LIST XH YH))
  BP (LIST (+ XH XP) (+ YH YP HH))
  T1 (POLAR BT PI (/ LH 2))
  T2 (POLAR T1 (/ PI 2) HH)
  T3 (POLAR T2 0 LH)
  T4 (POLAR T1 0 LH)
)
(SETQ T12 (POLAR T1 (/ PI 2) (/ HH 2))
  T34 (POLAR T4 (/ PI 2) (/ HH 2))
)
(COMMAND
  "PLINE" T4 "W" 0.1 0.1 T1 "ARC" T2 "LINE" T3 "ARC" T4 ""
```

```

"CIRCLE" T12 (- (/ HH 2) 5)
"CIRCLE" T34 (- (/ HH 2) 5)
)
)
(GUSH 120 10 200 30 0 10 )           ; вызов функции
(COMMAND "ZOOM" "A")

```

Параметрическое изображение платформы экскаватора представлено на рис. 3.18.

3.6.5. Программа параметрического изображения поворотной платформы экскаватора

```

; *****
; (POVPL LZ LP LD HD HP HK LK XS YS XC1 YC1) - функция
; создания параметрического изображения поворотной
; платформы экскаватора
; А р г у м е н т ы  ф у н к ц и и:
; XP - координата x базовой точки поворотной платформы
; YP - координата y базовой точки поворотной платформы

```

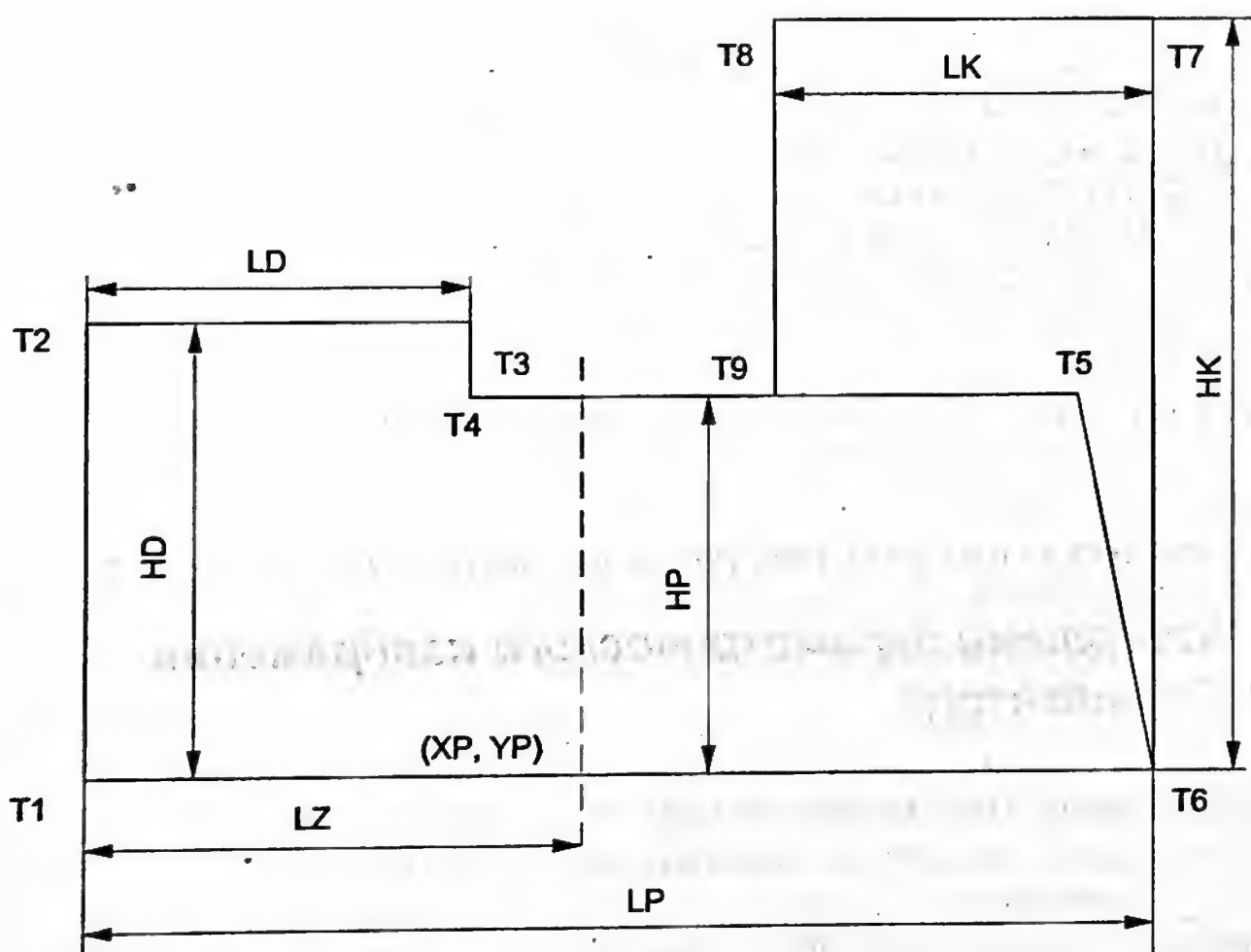


Рис. 3.18. Параметрическое изображение платформы экскаватора

```

; LZ - длина задней части платформы
; LP - длина поворотной платформы
; LD - длина двигательного отделения
; HD - высота задней части платформы
; HP - высота передней части платформы
; HK - высота кабины
; LK - длина кабины
; XS - координата x базовой точки пяты стрелы
; YS - координата y базовой точки пяты стрелы
; XC1 - координата x базовой точки крепления штока цилиндра стрелы
; YC1 - координата y базовой точки крепления штока цилиндра стрелы
; *****
(DEFUN POVPL (LZ LP LD HD HP HK LK XS YS XC1 YC1)
  (SETQ T1 (POLAR BP PI LZ)
    T2 (POLAR T1 (/ PI 2) HD)
    T3 (POLAR T2 0 LD)
    T4 (POLAR T3 (* 1.5 PI) (- HD HP))
    T5 (POLAR T4 0 (-(- LP LD) 10))
    T6 (POLAR BP 0 (- LP LZ))
    T7 (POLAR T6 (/ PI 2) HK)
    T8 (POLAR T7 PI LK)
    T9 (POLAR T8 (* 1.5 PI) (- HK HP))
    BS (POLAR BP 0 XS)
    BS (POLAR BS (/ PI 2) YS)
    BC1 (POLAR BP 0 XC1)
    BC1 (POLAR BC1 (/ PI 2) YC1))
  (COMMAND "PLINE" T6 "W" 0.1 0.1 T1 T2 T3 T4 T5 "C"
    "PLINE" T6 "W" 0.1 0.1 T7 T8 T9 "")
)
(SETQ BP '(120 30)) ; установка базовой точки платформы
(POVPL 70 140 50 60 50 100 50 60 20 80 10) ; вызов функции
(COMMAND "ZOOM" "A")

```

Параметрическое изображение стрелы экскаватора дано на рис. 3.19.

3.6.6. Программа параметрического изображения стрелы экскаватора

```

; *****
; (STRELA HS LHS LCS LCR HCR HSI1 LVS HSI2 HSR LSI1 LSI2
; US1 US) - функция создания параметрического изображения
; стрелы экскаватора
; А р г у м е н т ы   ф у н к ц и и :
; HS - ширина нижнего колена стрелы у опоры
; LHS - длина нижнего колена стрелы

```

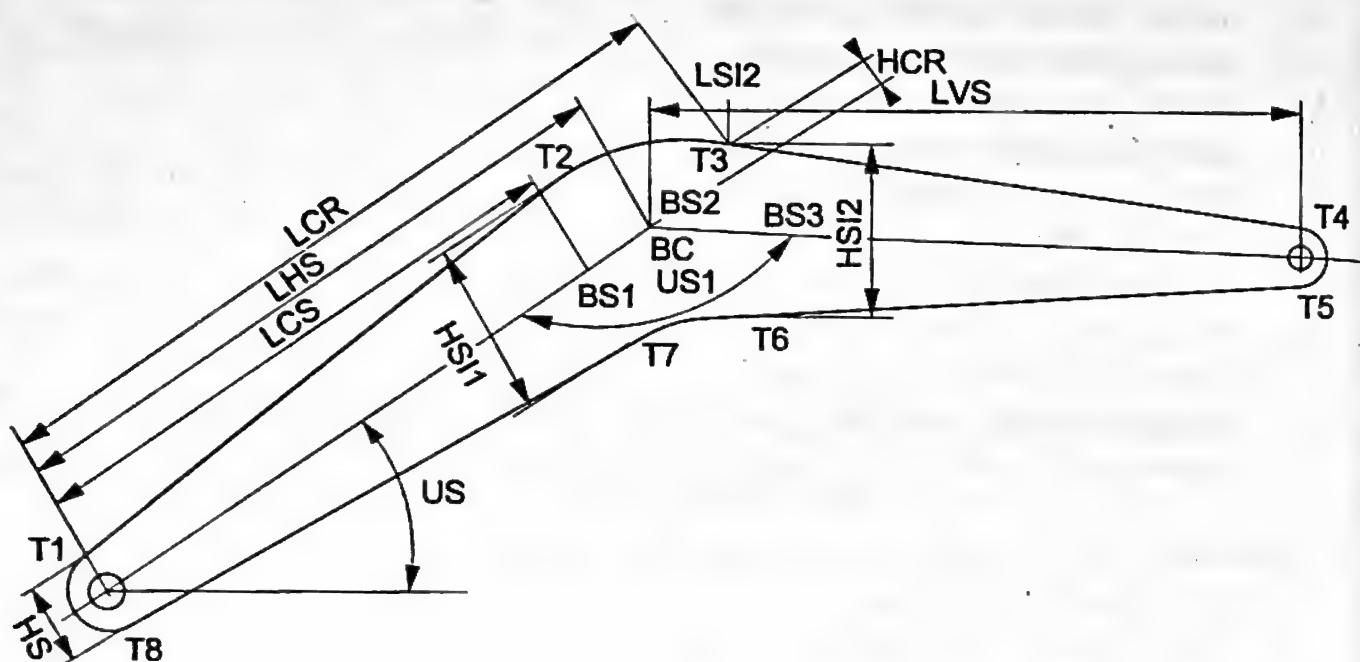



Рис. 3.19. Параметрическое изображение стрелы экскаватора

; LCS - расстояние от пята стрелы до шарнира крепления
 ; цилиндра стрелы
 ; LCR - расстояние от пята до оси шарнира крепления
 ; цилиндра рукояти
 ; HCR - расстояние от оси нижнего колена стрелы до шарнира
 ; крепления гидроцилиндра рукояти
 ; HSI1 - ширина нижнего колена стрелы у изгиба
 ; LVS - длина верхнего колена стрелы
 ; HSI2 - ширина верхней части стрелы у изгиба
 ; HSR - ширина верхней части стрелы у рукояти
 ; LSI1 - длина изогнутой части нижнего колена стрелы
 ; LSI2 - длина изогнутой части верхнего колена стрелы
 ; US1 - угол между верхней и нижней частями стрелы
 ; US - угол наклона нижней части стрелы
 ; *****
 (DEFUN STRELA (HS LHS LCS LCR HCR HSI1 LVS HSI2 HSR LSI1 LSI2 US1 US)
 (SETQ BC (POLAR BS US LCS)
 T1 (POLAR BS (+ US (/ PI 2)) (/ HS 2))
 T8 (POLAR BS (+ US (* 1.5 PI)) (/ HS 2))
 BS1 (POLAR BS US LHS)
 T2 (POLAR BS1 (+ US (/ PI 2)) (/ HSI1 2))
 T7 (POLAR BS1 (+ US (* 1.5 PI)) (/ HSI1 2))
 BS2 (POLAR BS1 US LSI1)
 US2 (- (+ US US1) PI)
 BS3 (POLAR BS2 US2 LSI2)
 T3 (POLAR BS3 (+ US2 (/ PI 2)) (/ HSI2 2))

```

T6 (POLAR BS3 (+ US2 (* 1.5 PI)) (/ HSI2 2))
BSR (POLAR BS3 US2 LVS)
T4 (POLAR BSR (+ US2 (/ PI 2)) (/ HSR 2))
T5 (POLAR BSR (+ US2 (* 1.5 PI)) (* 0.5 HSR))
)
(SETQ T9 (POLAR BS US LCR)
  BCR (POLAR T9 (+ US (/ PI 2)) HCR)
)
(COMMAND "PLINE" T1 "W" 0.1 0.1 T2 "ARC" T3 "LINE" T4
  "ARC" T5 "LINE" T6 "ARC" T7 "LINE" T8 "ARC" T1 ""
  "CIRCLE" BS (* 0.2 HS) "CIRCLE" BSR (* 0.2 HSR)
)
)
(SETQ BS '(30 30)) ; установка базовой точки стрелы
(STRELA 15 100 110 130 5 30 100 30 10 10 10 2.5 0.6) ; вызов функции
(COMMAND "ZOOM" "A")

```

Параметрическое изображение рукояти экскаватора представлено на рис. 3.20.

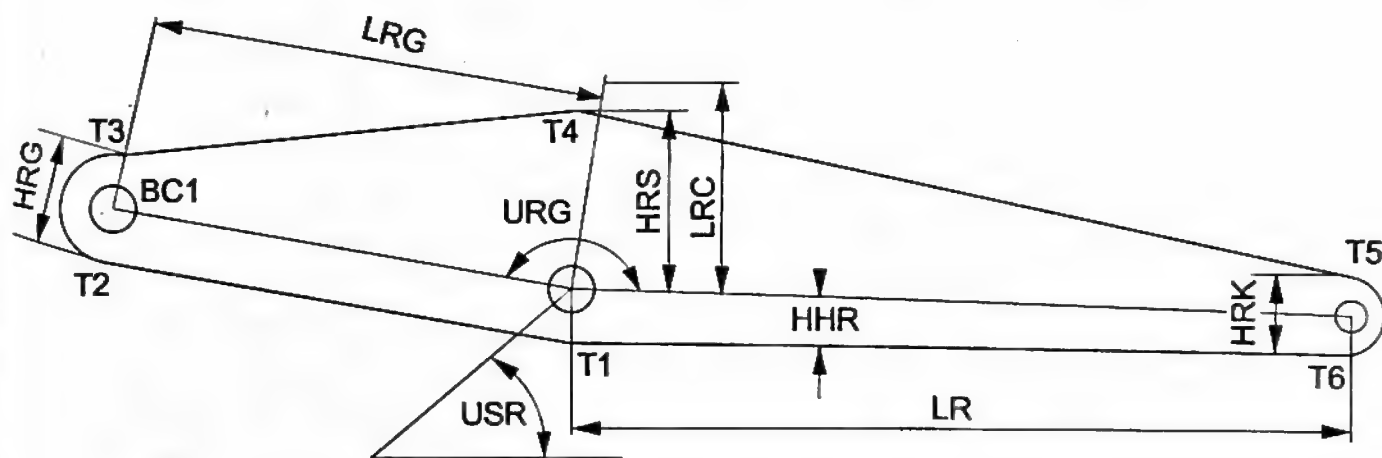


Рис. 3.20. Параметрическое изображение рукояти экскаватора

3.6.7. Программа параметрического изображения рукояти экскаватора

```

; *****
; (RUKO LR HRS HRK USR HHR LRG URG HRG US2 LRC URC) -
; функция создания параметрического изображения рукояти
; экскаватора
; А р г у м е н т ы  ф у н к ц и и:
; LR - длина рукояти
; HRS - ширина рукояти у стрелы
; HRK - ширина рукояти у ковша

```

```

; USR - угол между стрелой и рукоятью
; HHR - расстояние от шарнира до низа
; LRG - расстояние до крепления штока
; URG - угол крепления штока гидроцилиндра
; HRG - ширина рукояти в месте крепления штока
; US2 - угол наклона рукояти к стреле
; LRC - расстояние от шарнира до крепления гидроцилиндра ковша
; URC - угол крепления гидроцилиндра ковша
; *****
(DEFUN RUKO (LR HRS HRK USR HHR LRG URG HRG US2 LRC URC)
  (SETQ UR (+ US2 PI USR)
    BRK (POLAR BSR UR LR)
    T1 (POLAR BSR (- UR (/ PI 2)) HHR)
    T4 (POLAR BSR (- UR (* 1.5 PI)) (- HRS HHR))
    BC1 (POLAR BSR (- URG (- (* 2.0 PI) UR)) LRG)
    T2 (POLAR BC1 (- URG (- (* 1.5 PI) UR)) (/ HRG 2))
    T3 (POLAR BC1 (- URG (- (* 2.5 PI) UR)) (/ HRG 2))
    T5 (POLAR BRK (- UR (* 1.5 PI)) (* 0.5 HRK))
    T6 (POLAR BRK (- UR (* 0.5 PI)) (* 0.5 HRK))
    BC BCR
    LRC0 LRC
    URC0 URC)
  (COMMAND "PLINE" T3 "W" 0.1 0.1 T4 T5
    "ARC" T6 "LINE" T1 T2
    "ARC" T3 ""
    "CIRCLE" BC1 (* 0.2 HRG)
    "CIRCLE" BRK (* 0.2 HRK)
    "CIRCLE" BSR (* 0.1 HRS)
  )
)
(SETQ BSR '(80 30)) ; установка базовой точки рукояти
(RUKO 100 30 10 2.5 7 60 3.0 15 0.61 35 1.5) ; вызов функции
(COMMAND "ZOOM" "A")

```

Параметрическое изображение ковша представлено на рис. 3.21.

3.6.8. Программа параметрического изображения ковша экскаватора

```

; *****
; (KOV LK URK HK LZU HZ HG RK) - функция создания
; параметрического изображения ковша
; А р г у м е н т ы   ф у н к ц и и :
; LK - длина ковша по оси X

```

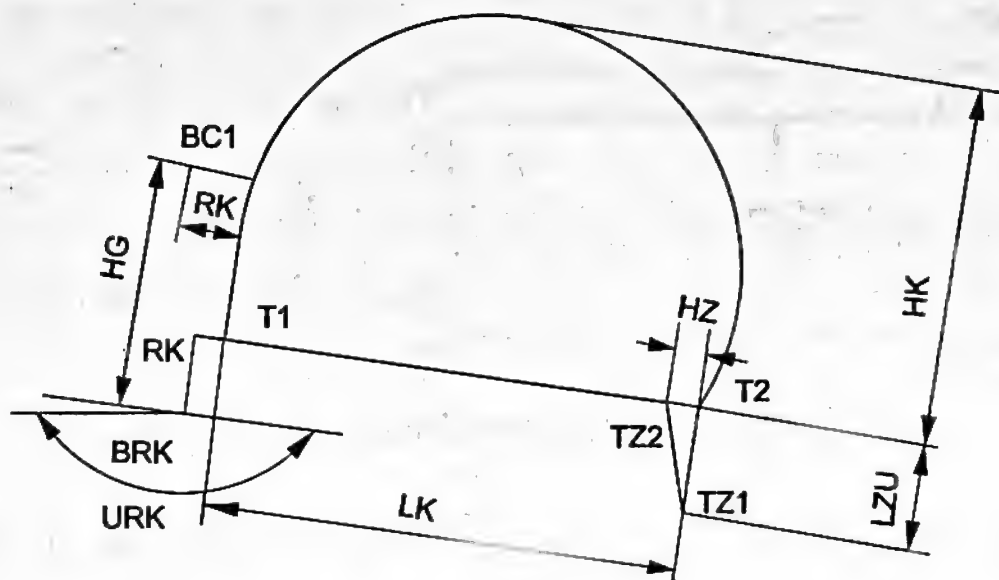


Рис. 3.21. Параметрическое изображение ковша

```
; URK - угол между рукоятью и ковшом
; HK - высота ковша
; LZU - длина зуба
; HZ - ширина основания зуба
; HG - расстояние от крепления ковша до цилиндра
; RK - расстояние от крепления ковша до линии T1-T2
; *****
(DEFUN KOV (LK URK HK LZU HZ HG RK)
  (SETQ UK (- (+ UR URK) PI)
    T1 (POLAR BRK UK RK)
    TZ (POLAR BRK UK LK)
    TZ1 (POLAR TZ (- UK (/ PI 2)) LZU)
    TZ2 (POLAR BRK UK (- LK HZ))
    TC1 (POLAR T1 (- UK (* 1.5 PI)) HG)
    BC1 TC1
    BC (POLAR BSR (+ UK URC0) LRC0)
  )
  (COMMAND
    "PLINE" T1 "W" 0.1 0.1 TC1 "ARC" TZ "LINE" TZ1 TZ2 BRK "")
  )
  (SETQ UR 0.0
    URC0 30
    BSR '(80 30)
    LRC0 1.5
    BRK '(20 40))
  (KOV 50 3.0 40 10 3 10 5)
  (COMMAND "ZOOM" "A")
```

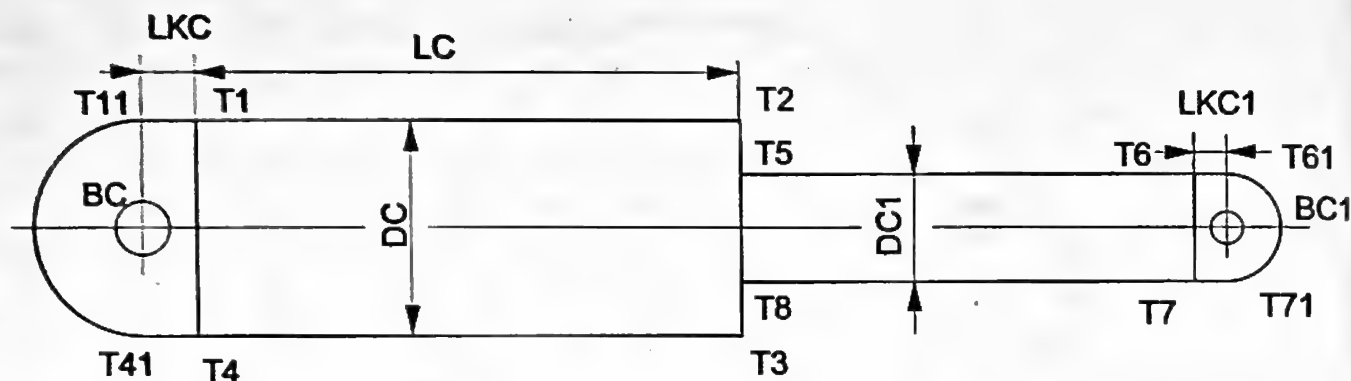



Рис. 3.22. Параметрическое изображение гидроцилиндра

Параметрическое изображение гидроцилиндра представлено на рис. 3.22.

3.6.9. Программа параметрического изображения гидроцилиндра

```

;*****
; (GCL LC LKC DC DC1 LKC1) - функция создания
; параметрического изображения гидроцилиндра
; А р г у м е н т ы  ф у н к ц и и:
; BC - точка крепления гидроцилиндра
; BC1 - точка крепления штока гидроцилиндра
; LC - длина гидроцилиндра
; LKC - длина крепления цилиндра
;* DC - диаметр гидроцилиндра
; DC1 - диаметр штока цилиндра
; LKC1 - длина крепления штока
;*****
(DEFUN GCL (LC LKC DC DC1 LKC1)
  (SETQ UC (ANGLE BC BC1)
    PI05 (* PI 0.5)
    TC (POLAR BC UC LKC)
    T1 (POLAR TC (- UC (* 1.5 PI)) (/ DC 2))
    T11 (POLAR T1 (+ UC PI) LKC)
    T2 (POLAR T1 UC LC)
    T4 (POLAR TC (- UC PI05) (/ DC 2))
    T41 (POLAR T4 (+ UC PI) LKC)
    T3 (POLAR T4 UC LC)
    TCL (POLAR TC UC LC)
    T5 (POLAR TCL (- UC (* 1.5 PI)) (/ DC1 2))
    T8 (POLAR TCL (- UC PI05) (/ DC1 2))
    LC1 (- (DISTANCE BC BC1) LKC LC LKC1)
    T6 (POLAR T5 UC LC1)

```

```

T61 (POLAR T6 UC LKC1)
T7 (POLAR T8 UC LC1)
T71 (POLAR T7 UC LKC1)
)
(COMMAND
  "PLINE" T1 "W" 0.1 0.1 T2 T3 T4 T1 T11 "ARC" T41 "LINE" T4 ""
  "PLINE" T6 "W" 0.1 0.1 T5 T8 T7 T6 T61 "ARC" T71 "LINE" T7 ""
  "CIRCLE" BC (* LKC 0.5) "CIRCLE" BC1 (* LKC1 0.5))
)
(SETQ BC '(30 30) ; точка крепления гидроцилиндра
      BC1 '(130 30)) ; и штока гидроцилиндра
(GCL 50 5 20 10 3) ; вызов функции
(COMMAND "ZOOM" "A")

```

Координаты точек BC и BC1 определяются до вызова функции создания параметрического изображения гидроцилиндра.

3.6.10. Программа параметрического изображения общего вида экскаватора

```

; *****
; Комплекс функций для создания параметрического изображения
; общего вида одноковшового гидравлического экскаватора
; (GCL...) - функция параметрического изображения гидроцилиндра
; (KOV...) - функция параметрического изображения ковша
; (RUKO...) - функция параметрического изображения
; рукояти экскаватора
; (STRELA...) - функция параметрического изображения
; стрелы экскаватора
; (GUSH...) - функция параметрического изображения
; гусеничного хода
; (POVPL...) - функция параметрического изображения
; поворотной платформы
; *****
(DEFUN ECSAG ()
  (SETVAR "CMDECHO" 0)
  (SETVAR "BLIPMODE" 0)
  (GUSH 120 10 200 30 0 10 )
  (POVPL 100 180 50 60 50 100 50 60 20 80 10)
  (STRELA1 15 100 110 130 5 25 100 25 10 10 10 2.5 1.2)
  (GCL 50 5 10 5 3)
  (RUKO 100 30 10 2.5 7 60 3.0 15 0.41 35 1.5 )
  (GCL 50 5 10 5 3)

```

```
(KOV 50 2.4 40 10 3 10 5)  
(GCL 50 5 10 5 3)  
(COMMAND "ZOOM" "E")  
(PRIN1)  
)  
(ECSAG) ; вызов функции
```

После вызова функции **(ECSAG)** на экране появится параметрическое изображение общего вида гидравлического экскаватора (рис. 3.23).

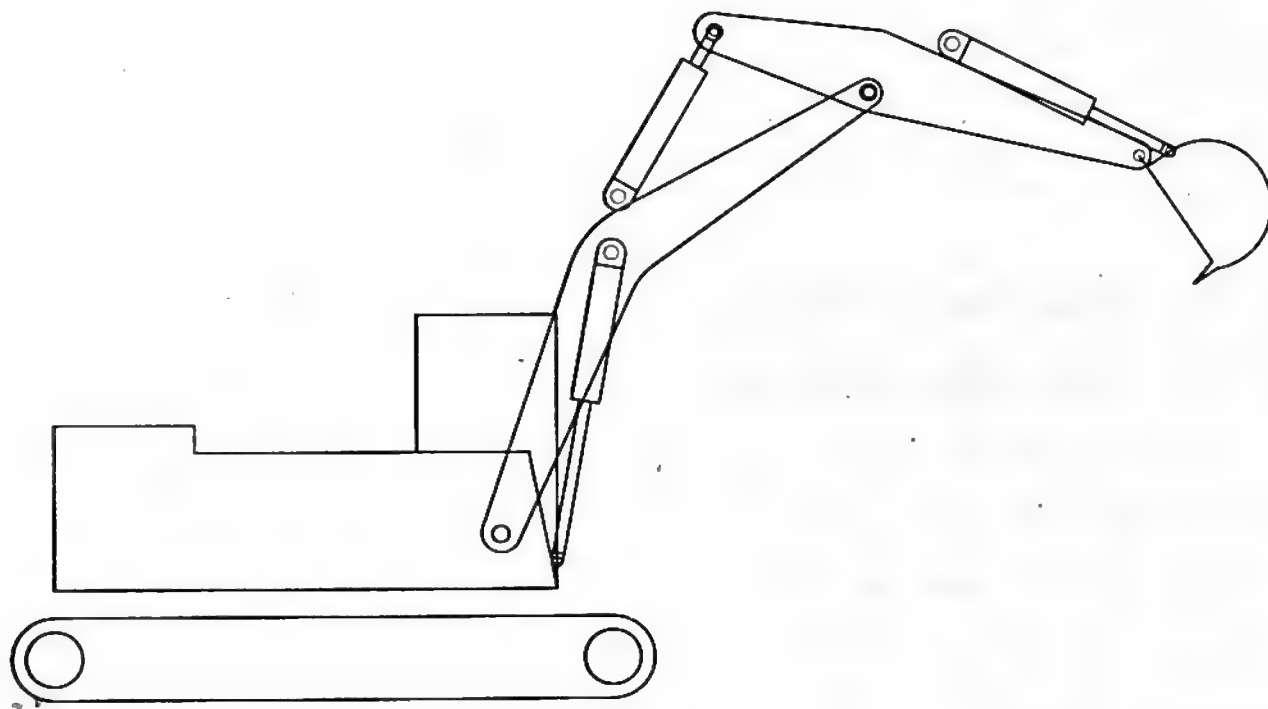


Рис. 3.23. Результат выполнения функции (ECSAG)

Глава 4

СОЗДАНИЕ ФРАГМЕНТОВ СИСТЕМ ОБРАБОТКИ ИНФОРМАЦИИ

Преобразование алгебраических выражений	162
Обработка статистических данных	176
Разработка простейшей реляционной базы данных	185

В этой главе рассматриваются процессы создания систем обработки информации, представленной в виде символов и чисел, в том числе аналитических преобразований алгебраических выражений. Здесь же говорится об обработке статистических данных и их графическом изображении, создании простейших баз данных и др.

4.1. Преобразование алгебраических выражений

Рассмотрим основные этапы создания системы преобразования алгебраических выражений. Напомним, что алгебраическое выражение – это выражение, составленное из букв (символов) и чисел, соединенных знаками действий сложения, вычитания, умножения, деления, возведения в целую степень и извлечения корня (показатели степени и корня должны быть постоянными числами). Если некоторые из букв (или все) считать переменными, то алгебраическое выражение есть алгебраическая функция. Ограничимся исследованием алгебраического выражения с одной переменной и алгебраической функции с одной переменной.

4.1.1. Постановка задачи

Разработать фрагменты системы для автоматизации обработки алгебраических выражений, представленных в виде символов и чисел. Исходные и конечные выражения должны быть представлены в обычном для пользователя виде (в инфиксной форме). В процессе вычислений необходимо предусмотреть упрощение алгебраических выражений, использование в алгебраических выражениях, кроме операций сложения, вычитания, умножения и деления, операций возведения в целую степень и дифференцирования.

Необходимо обеспечить возможность ввода нескольких исходных аналитических выражений в одной строке.

4.1.2. Выявление основных особенностей, взаимосвязей и количественных закономерностей

Будем считать, что все составляющие алгебраических выражений отделяются друг от друга пробелом. Разработаем функцию аналитических преобразований (ANALITIC), которая будет считывать исходные алгебраические выражения из файла, например, ANALITIC.DAN. Результаты вычисления и преобразования алгебраических выражений должны выводиться в файл, например ANALITIC.REZ. В качестве знака операции возведения в степень будем использовать знак ^, операцию дифференцирования обозначим символом D, операцию присваивания – знаком =. Отделять аналитические выражения будем с помощью знака : (двоеточия).

Установим следующий порядок выполнения действий над аналитическими выражениями: / * – + = : D.

Для упрощения программы будем использовать бинарные операции, то есть операции с двумя аргументами. В таком случае отрицательное значение символа, например, $-S$ следует записывать в виде $0 - S$.

4.1.3. Разработка алгоритмов и комплекса функций для преобразования аналитических выражений

Рассмотрим вначале алгоритм преобразования алгебраических выражений из инфиксной формы в префиксную. Алгебраические выражения записываются, как правило, в инфиксной форме, например:

$A * 5 + B$

Язык AutoLISP работает с выражениями в префиксной форме, то есть выражения имеют следующий вид:

$(* A (+ 5 B))$

Для преобразования используем промежуточный буфер – стек, в котором сохраняется знак операции. Первоначально стек пустой (NIL) и выражения в префиксной форме – (NIL) не существует. Из выражения в инфиксной форме слева направо последовательно выбираем элементы выражения (символы, числа, знаки операций) и переносим в выражение в префиксной форме. При этом время от времени используем стек:

- если элемент – символ или число, его помещают прямо в список справа (выражение в префиксной форме);
- если элемент – знак операции, его помещают или в список справа (выражение в префиксной форме) или в стек:
 - если приоритет перемещаемого знака операции выше приоритета знака операции, находящегося в верхней ячейке стека, знак операции перемещается в стек;
 - если приоритет перемещаемого знака операции ниже или равен приоритету знака операции, находящегося в верхней ячейке стека, префиксная форма формируется следующим образом: первым элементом списка становится первый элемент стека, затем второй и первый элементы списка в префиксной форме, вслед за ними остальные элементы.
- если выражение пусто, а стек не пустой, префиксная форма формируется так: первым элементом списка становится первый элемент стека, затем второй и первый элементы списка справа (выражение в префиксной форме), затем остальные элементы.

В стеке накапливаются знаки операций, начиная с наиболее низких приоритетов и кончая высшими.

После того как все символы и числа будут перенесены в список справа (выражение в префиксной форме), начнем последовательный перенос знаков операций из стека. При этом знаки операций будем извлекать из стека по принципу «последний пришел, первый вышел».

Для определения приоритета операций создадим функцию (OLD...).

```
(SETQ ZNAK '(/ * - + = : D)) ; приоритет операций
(DEFUN OLD (OP1 OP2) ; определение приоритета операций
  (OR (NULL OP2) (MEMBER OP2 (MEMBER OP1 ZNAK)))
)
```

Разработаем функцию (**INFPREF...**) преобразования аналитического выражения *S* из инфиксной формы в префиксную *SP*, используя промежуточный буфер – стек *ST*.

```
(DEFUN INFPREF (SP ST S)
  (COND
    ((NULL S) (IF (CAR ST) (PR SP ST S) (CAR SP)))
    ((ATOM S) S)
    ((ATOM (CAR S))
      (COND
        ((NOT (MEMBER (CAR S) ZNAK))
          (INFPREF (CONS (CAR S) SP) ST (CDR S)))
        ((OLD (CAR S) (CAR ST))
          (INFPREF SP (CONS (CAR S) ST) (CDR S)))
        (T (PR SP ST S)))
      )
    (T (INFPREF (CONS (INFPREF NIL ' (NIL) (CAR S)) SP) ST (CDR S))))
)
(DEFUN PR (SP ST S)
  * (INFPREF (CONS (LIST (CAR ST) (CADR SP) (CAR SP))
    (CDDR SP)) (CDR ST) S)
)
```

Далее разработаем функции для выполнения основных операций с выражениями, представленными в виде символов и чисел. Используя рекуррентную формулу $X^n = X * X^{n-1}$, определим правила возведения выражения в целое число раз:

```
(DEFUN ^ (X Y) ; функция возведения X в целое число раз
  (COND ((EQ Y 0) 1)
    (T (* X (^ X (- Y 1)))))
)
```

Правила сложения выражений, представленных в виде символов и чисел:

- если *X* – число и оно равно нулю, то сумма *X* и *Y* равна *Y*;
- если *Y* – число и оно равно нулю, то сумма *X* и *Y* равна *X*;
- если *X* и *Y* – числа, то их сумма равна $(+ X Y)$;
- если *X* и *Y* – списки, то необходимо провести вычисление их, а затем сформировать обращение к функции сложения;
- если *X* – список, то необходимо провести вычисление его, а затем сформировать обращение к функции сложения;

- если Y – список, то необходимо провести вычисление его, а затем сформировать обращение к функции сложения;
- если все вышеперечисленные условия не выполняются, следует сформировать обращение к функции сложения.

Соответствующая функция сложения выражений, представленных в виде символов и чисел, будет выглядеть следующим образом:

```
(DEFUN U+ (X Y)
  (COND ((AND (NUMBERP X) (ZEROP X)) Y)
        ((AND (NUMBERP Y) (ZEROP Y)) X)
        ((AND (NUMBERP X) (NUMBERP Y)) (+ X Y))
        ((AND (LISTP X) (LISTP Y))
         (LIST '+ (EVALS X) (EVALS Y)))
        ((LISTP X) (LIST '+ (EVALS X) Y))
        ((LISTP Y) (LIST '+ X (EVALS Y)))
        (T (LIST '+ X Y)))
  )
)
```

Правила вычитания выражений, представленных в виде символов и чисел:

- если X и Y равны, то результатом вычитания будет 0;
- если Y – число и оно равно нулю, то результат будет равен X ;
- если X и Y – числа, то их разность равна $(- X Y)$;
- если X и Y – списки, то необходимо провести вычисление их, а затем сформировать обращение к функции вычитания;
- если X – список, то необходимо провести вычисление его, а затем сформировать обращение к функции вычитания;
- если Y – список, то необходимо провести вычисление его, а затем сформировать обращение к функции вычитания;
- если все вышеперечисленные условия не выполняются, следует сформировать обращение к функции вычитания.

Соответствующая функция вычитания выражений, представленных в виде символов и чисел, будет выглядеть следующим образом:

; функция вычитания символьных выражений

```
(DEFUN U- (X Y)
  (COND ((EQUAL X Y) 0)
        ((AND (NUMBERP Y) (ZEROP Y)) X)
        ((AND (NUMBERP X) (NUMBERP Y)) (- X Y))
        ((AND (LISTP X) (LISTP Y))
         (LIST '- (EVALS X) (EVALS Y)))
        ((LISTP X) (LIST '- (EVALS X) Y))
        ((LISTP Y) (LIST '- X (EVALS Y)))
  )
)
```



```
(T (LIST '- X Y)))
```

```
)
```

Правила умножения выражений, представленных в виде символов и чисел:

- если X равен 1, то результатом умножения будет Y ;
- если Y равен 1, то результатом умножения будет X ;
- если X или Y — число и оно равно нулю, то результат будет равен 0;
- если X и Y — числа, то результат равен $(* X Y)$;
- если X и Y — списки и они равны, то сначала необходимо провести вычисление X , а затем сформировать обращение к функции возведения в степень;
- если X и Y — списки, то необходимо провести вычисление X и Y , а затем сформировать обращение к функции умножения;
- если X — список, следует провести его вычисление, а затем сформировать обращение к функции умножения;
- если Y — список, то необходимо провести его вычисление, а затем сформировать обращение к функции умножения;
- если X равен Y , то необходимо сформировать обращение к функции возведения в степень;
- если все вышеперечисленные условия не выполняются, следует сформировать обращение к функции умножения.

Соответствующая функция умножения выражений, представленных в виде символов и чисел, будет выглядеть следующим образом:

; функция умножения выражений, представленных в виде

; символов и чисел

```
(DEFUN U* (X Y)
  (COND ((EQUAL X 1) Y)
        ((EQUAL Y 1) X)
        ((OR (AND (NUMBERP X) (ZEROP X))
              (AND (NUMBERP Y) (ZEROP Y))) 0)
        ((AND (NUMBERP X) (NUMBERP Y)) (* X Y))
        ((AND (LISTP X) (LISTP Y) (EQUAL X Y))
         (LIST '^ (EVALS X) 2))
        ((AND (LISTP X) (LISTP Y))
         (LIST '* (EVALS X) (EVALS Y)))
        ((LISTP X) (LIST '* (EVALS X) Y))
        ((LISTP Y) (LIST '* X (EVALS Y)))
        ((EQUAL X Y) (LIST '^ X 2))
        (T (LIST '* X Y)))
```

```
)
```

```
)
```

Правила деления выражений, представленных в виде символов и чисел:

- если X – число и оно равно 0, то результат деления равен 0;
- если Y – число и оно равно 0, то результат деления равен 'inf';
- если Y равен 1, то результатом деления будет X ;
- если X и Y равны, то результат равен 1;
- если X и Y – числа, то результат равен $(/ X Y)$;
- если X и Y – списки, то необходимо провести вычисление X и Y , а затем сформировать обращение к функции деления;
- если X – список, то необходимо провести его вычисление, а затем сформировать обращение к функции деления;
- если Y – список, то необходимо провести его вычисление, а затем сформировать обращение к функции деления;
- если все вышеперечисленные условия не выполняются, следует сформировать обращение к функции деления.

Соответствующая функция деления выражений, представленных в виде символов, будет выглядеть следующим образом:

; функция деления выражений, представленных в виде

; символов и чисел

```
(DEFUN U/ (X Y)
  (COND ((AND (NUMBERP X) (ZEROP X)) 0)
        ((AND (NUMBERP Y) (ZEROP Y)) "INF")
        ((EQUAL Y 1) X)
        ((EQUAL X Y) 1)
        ((AND (NUMBERP X) (NUMBERP Y)) (/ X Y))
        ((AND (LISTP X) (LISTP Y))
         (LIST '/ (EVALS X) (EVALS Y)))
        ((LISTP X) (LIST '/ (EVALS X) Y))
        ((LISTP Y) (LIST '/ X (EVALS Y)))
        (T (LIST '/ X Y)))
)
```

Правила возведения в целую степень выражений, представленных в виде символов и чисел:

- если X равен 1, то результат возведения в степень равен 1;
- если Y равен 1, то результат возведения в степень равен X ;
- если X – число и оно равно нулю, то результат будет равен 0;
- если Y – число и оно равно нулю, то результат будет равен 1;
- если X и Y – числа, то результат равен $(^ X Y)$;
- если X и Y – списки, необходимо провести их вычисление, а затем сформировать обращение к функции возведения в степень;
- если X – список, необходимо провести его вычисление, а затем сформировать обращение к функции возведения в степень;

- если Y – список, необходимо провести его вычисление, а затем сформировать обращение к функции возведения в степень;
- если все вышеперечисленные условия не выполняются, следует сформировать обращение к функции возведения в степень.

Соответствующая функция возведения в степень выражений, представленных в виде символов и чисел, будет выглядеть так;

```
(DEFUN U^ (X Y) ; функция возведения в целую степень
  (COND ((EQUAL X 1) 1)
        ((EQUAL Y 1) X)
        ((EQUAL Y 0) 1)
        ((AND (NUMBERP X) (ZEROP X)) 0)
        ((AND (NUMBERP Y) (ZEROP Y)) 1)
        ((AND (NUMBERP X) (NUMBERP Y)) (^ X Y))
        ((AND (LISTP X) (LISTP Y))
         (LIST '^ (EVALS X) (EVALS Y)))
        ((LISTP X) (LIST '^ (EVALS X) Y))
        ((LISTP Y) (LIST '^ X (EVALS Y)))
        (T (LIST '^ X Y)))
  )
)
```

Функции деления выражений и присваивания будут выглядеть так:

```
(DEFUN U: (X Y) ; функция деления выражений
  Y)
(DEFUN U= (X Y) ; функция присваивания
  (IF (BOUND P 'X)
    (IF (ASSOC X LXY)
      (SETQ LXY (SUBST (LIST X Y) (ASSOC X LXY) LXY))
      (SETQ LXY (CONS (LIST X Y) LXY)))
    (ERROR "~%~S НЕЛЬЗЯ ПРИСВОИТЬ ЗНАЧЕНИЕ" X ))
  )
)
```

Определим функцию (EVALD...) для вычисления выражений символов, представленных в виде дерева в префиксной форме, и функцию (EVALV...) для вычисления ветвей дерева.

Предварительно создадим ассоциативный список FN в виде:

```
'((+ U+) (- U-) (* U*) (/ U/) (^ U^) (= U=) (: U:))
(SETQ FN (MAPCAR 'CONS '(+ - * / ^ = :)
  '(U+ U- U* U/ U^ U= U:)))
)
```

; функция вычисления дерева – бинарного списка в префиксной форме

```
(DEFUN EVALD (X / Y1)
  (COND ((NUMBERP X) X)
```

```

((ATOM X) (IF (SETQ Y1 (CADR (ASSOC X LXY)))
              (EVALD Y1) X))
(T (EVALV (NTH 0 X) (MAPCAR (FUNCTION EVALD) (CDR X))))))
)
; функция вычисления ветвей дерева
(DEFUN EVALV (OPER ARGS / OP)
  (SETQ OP (CDR (ASSOC OPER FN)))
  (IF OP (APPLY OP ARGS) (LIST OP (EVALD (NTH 0 ARGS))
                                   (EVALD (NTH 1 ARGS))))))
)

```

Функция преобразования префиксной формы выражений в инфиксную и функция удаления лишних скобок представлены ниже:

```

(DEFUN PREINF (S) ; функция преобразования S
                  ; в инфиксную форму
  (IF (ATOM S) S (APPEND (DELOP (NTH 0 S) (NTH 1 S))
                        (LIST (NTH 0 S))
                        (DELOP (NTH 0 S) (NTH 2 S)))))
)
(DEFUN DELOP (OP S / S1) ; функция удаления лишних скобок
  (SETQ S1 (PREINF S))
  (IF (OR (ATOM S1) (OLD OP (NTH 1 S1)))
      (LIST S1) S1))
)

```

Программа преобразования аналитических выражений представлена ниже.

4.1.4. Программа преобразования аналитических выражений

```

; *****
; Программа преобразования аналитических выражений ANALITIC
; *****
; Возможные формы записи выражений в файле ANALITIC.DAN
; A = 4 | или A = 4 : C = 6 : B = C ^ A : E = C + B / E
; C = 6 |
; B = C ^ A
; E = C + B / E
; Результат преобразования записывается в файл ANALITIC.REZ в виде:
; (6 + 1296 / E)
;
; Перечень используемых функций
; (INFPREF SP ST S1) - функция преобразования инфиксной формы
; записи выражения S1 в префиксную SP в виде бинарного списка,

```



```

; например, выражение '(C + A * D) преобразуется в '(+ C (* A D))
; (OLD OP1 OP2) - функция определения приоритета операций
; (U+ X Y) - функция сложения выражений, представленных в виде
; символов и чисел
; (U- X Y) - функция вычитания выражений, представленных в виде
; символов и чисел
; (U^ X Y) - функция возведения выражения в целое число раз
; (U* X Y) - функция умножения выражений, представленных в виде
; символов и чисел
; (U/ X Y) - функция деления выражений, представленных
; в виде символов и чисел
; (EVALS X / OP) - функция упрощения алгебраического выражения
; (U: X Y) - функция разделения выражений
; (U= X Y) - функция присваивания
; (EVALS X) - функция вычисления выражений, представленных
; в виде символов
; (EVALD X) - функция вычисления дерева выражений
; (PREINF S) - функция преобразования выражения S
; в инфиксную форму
; (DELOP OP S) - функция удаления лишних скобок
; (EVALV OPER ARGS) - функция вычисления ветвей дерева
; (READL) - функция считывания строки в виде списка из файла
; *****
(SETQ ZNAK ' (^ / * - + = :)) ; список приоритета операций
(DEFUN INFPREF (SP ST S)
  (COND
    ((NULL S) (IF (CAR ST) (PR SP ST S) (CAR SP)))
    ((ATOM S) S)
    ((ATOM (CAR S))
      (COND
        ((NOT (MEMBER (CAR S) ZNAK))
          (INFPREF (CONS (CAR S) SP) ST (CDR S)))
        ((OLD (CAR S) (CAR ST))
          (INFPREF SP (CONS (CAR S) ST) (CDR S)))
        (T (PR SP ST S))))
    (T (INFPREF (CONS (INFPREF NIL ' (NIL) (CAR S)) SP)
      ST (CDR S))))
  )
(DEFUN PR (SP ST S)
  (INFPREF (CONS (LIST (CAR ST) (CADR SP) (CAR SP))
    (CDDR SP)) (CDR ST) S)
  )
; функция определения приоритета операций
(DEFUN OLD (OP1 OP2)

```

```
(OR (NULL OP2) (MEMBER OP2 (MEMBER OP1 ZNAK)))
)
```

```
(DEFUN ^ (X Y)
  (COND ((EQ Y 0) 1)
        (T (* X (^ X (- Y 1)))))
)
```

; функция сложения выражений, представленных в виде
; символов и чисел

```
(DEFUN U+ (X Y)
  (COND ((AND (NUMBERP X) (ZEROP X)) Y)
        ((AND (NUMBERP Y) (ZEROP Y)) X)
        ((AND (NUMBERP X) (NUMBERP Y)) (+ X Y))
        ((AND (LISTP X) (LISTP Y))
         (LIST '+ (EVALS X) (EVALS Y)))
        ((LISTP X) (LIST '+ (EVALS X) Y))
        ((LISTP Y) (LIST '+ (EVALS Y) X))
        (T (LIST '+ X Y)))
)
```

; функция вычитания выражений, представленных в виде
; символов и чисел

```
(DEFUN U- (X Y)
  (COND ((EQUAL X Y) 0)
        ((AND (NUMBERP Y) (ZEROP Y)) X)
        ((AND (NUMBERP X) (NUMBERP Y)) (- X Y))
        ((AND (LISTP X) (LISTP Y))
         (LIST '- (EVALS X) (EVALS Y)))
        ((LISTP X) (LIST '- (EVALS X) Y))
        ((LISTP Y) (LIST '- X (EVALS Y)))
        (T (LIST '- X Y)))
)
```

; функция возведения выражения в целое число раз

```
(DEFUN U^ (X Y)
  (COND ((EQUAL X 1) 1)
        ((EQUAL Y 1) X)
        ((EQUAL Y 0) 1)
        ((AND (NUMBERP X) (ZEROP X)) 0)
        ((AND (NUMBERP Y) (ZEROP Y)) 1)
        ((AND (NUMBERP X) (NUMBERP Y)) (^ X Y))
        ((AND (LISTP X) (LISTP Y))
         (LIST '^ (EVALS X) (EVALS Y)))
        ((LISTP X) (LIST '^ (EVALS X) Y)))
)
```

```

      ((LISTP Y) (LIST '^ X (EVALS Y)))
      (T (LIST '^ X Y))
    )
  )
; функция умножения выражений, состоящих из символов и чисел
(DEFUN U* (X Y)
  (COND ((EQUAL X 1) Y)
        ((EQUAL Y 1) X)
        ((OR (AND (NUMBERP X) (ZEROP X))
              (AND (NUMBERP Y) (ZEROP Y))) 0)
        ((AND (NUMBERP X) (NUMBERP Y)) (* X Y))
        ((AND (LISTP X) (LISTP Y) (EQUAL X Y))
         (LIST '^ (EVALS X) 2))
        ((AND (LISTP X) (LISTP Y))
         (LIST '* (EVALS X) (EVALS Y)))
        ((LISTP X) (LIST '* (EVALS X) Y))
        ((LISTP Y) (LIST '* X (EVALS Y)))
        ((EQUAL X Y) (LIST '^ X 2))
        (T (LIST '* X Y))
  )
)
; функция деления выражений, представленных в виде
; символов и чисел
(DEFUN U/ (X Y)
  (COND ((AND (NUMBERP X) (ZEROP X)) 0)
        ((AND (NUMBERP Y) (ZEROP Y)) "INF")
        ((EQUAL Y 1) X)
        ((EQUAL X Y) 1)
        ((AND (NUMBERP X) (NUMBERP Y)) (/ X Y))
        ((AND (LISTP X) (LISTP Y))
         (LIST '/ (EVALS X) (EVALS Y)))
        ((LISTP X) (LIST '/ (EVALS X) Y))
        ((LISTP Y) (LIST '/ X (EVALS Y)))
        (T (LIST '/ X Y))
  )
)
(SETQ FN (MAPCAR 'CONS '(+ - * / ^)
                  '(U+ U- U* U/ U^)))
; функция вычисления выражений, представленных в виде
; символов и чисел
(DEFUN EVALS (X / OP)
  (SETQ ARGS (CDR X)
            OP (CDR (ASSOC (CAR X) FN)))
  (APPLY OP ARGS)
)

```

; функция деления выражений

```
(DEFUN U: (X Y)
  Y)
```

; функция присваивания

```
(DEFUN U= (X Y)
  (IF (BOUNDP 'X)
    (IF (ASSOC X LXY)
      (SETQ LXY (SUBST (LIST X Y) (ASSOC X LXY) LXY))
      (SETQ LXY (CONS (LIST X Y) LXY)))
    (ERROR "~%~S НЕЛЬЗЯ ПРИСВОИТЬ ЗНАЧЕНИЕ" X )))
```

```
(SETQ FN (MAPCAR 'CONS ' (+ - * / ^ = : )
  '(U+ U- U* U/ U^ U= U:)))
```

; функция вычисления дерева - бинарного списка в префиксной форме

```
(DEFUN EVALD (X / Y1)
  (COND ((NUMBERP X) X)
    ((ATOM X) (IF (SETQ Y1 (CADR (ASSOC X LXY)))
      (EVALD Y1) X))
    (T (EVALV (NTH 0 X) (MAPCAR (FUNCTION EVALD) (CDR X))))))
```

; функция вычисления ветвей дерева

```
(DEFUN EVALV (OPER ARGS / OP)
  (SETQ OP (CDR (ASSOC OPER FN)))
  (IF OP (APPLY OP ARGS) (LIST OP (EVALD (NTH 0 ARGS))
    (EVALD (NTH 1 ARGS)))))
```

; функция преобразования выражения S в инфиксную форму

```
(DEFUN PREINF (S)
  (IF (ATOM S) S (APPEND (DELOP (NTH 0 S) (NTH 1 S))
    (LIST (NTH 0 S))
    (DELOP (NTH 0 S) (NTH 2 S)))))
```

; функция удаления лишних скобок

```
(DEFUN DELOP (OP S / S1)
  (SETQ S1 (PREINF S))
  (IF (OR (ATOM S1) (OLD OP (NTH 1 S1)))
    (LIST S1) S1))
```

; Открытие файла ANALITIC.DAN для считывания исходных данных

```
(SETQ FR (OPEN "C:\\ALISP\\ANALITIC.DAN" "r"))
```

; Открытие файла ANALITIC.REZ для записи результатов расчета

```
(SETQ FW (OPEN "C:\\ALISP\\ANALITIC.REZ" "w"))
```

; Создание функции (READL) для считывания строки в виде списка


```

(DEFUN READL () (STRCAT "( " Z " )")) (TEXTSCR)
(DEFUN ANALITIC (/ S)
  (SETQ LXY '())
  (WHILE (NOT (EQ (SETQ Z (READ-LINE FR)) NIL))
    (PRINT '<= )
    (SETQ S (READ (READL)))
    (PRINC S)
    (PRINT '<= FW)
    (PRINC S FW)
    (SETQ R (PREINF (CADR (CAR (EVALD (INFPREF NIL ' (NIL)
      S))))))
  )
  (PRINT '=> )
  (PRINC R)
  (PRINT '=> FW)
  (PRINC R FW)
  (CLOSE FR)
  (CLOSE FW)
)
(ANALITIC)

```

Пример ввода выражений, представленных в виде символов и чисел, в файл ANALITIC.DAN:

```

A = 4
C = 6
B = C ^ A
E = C + B / E

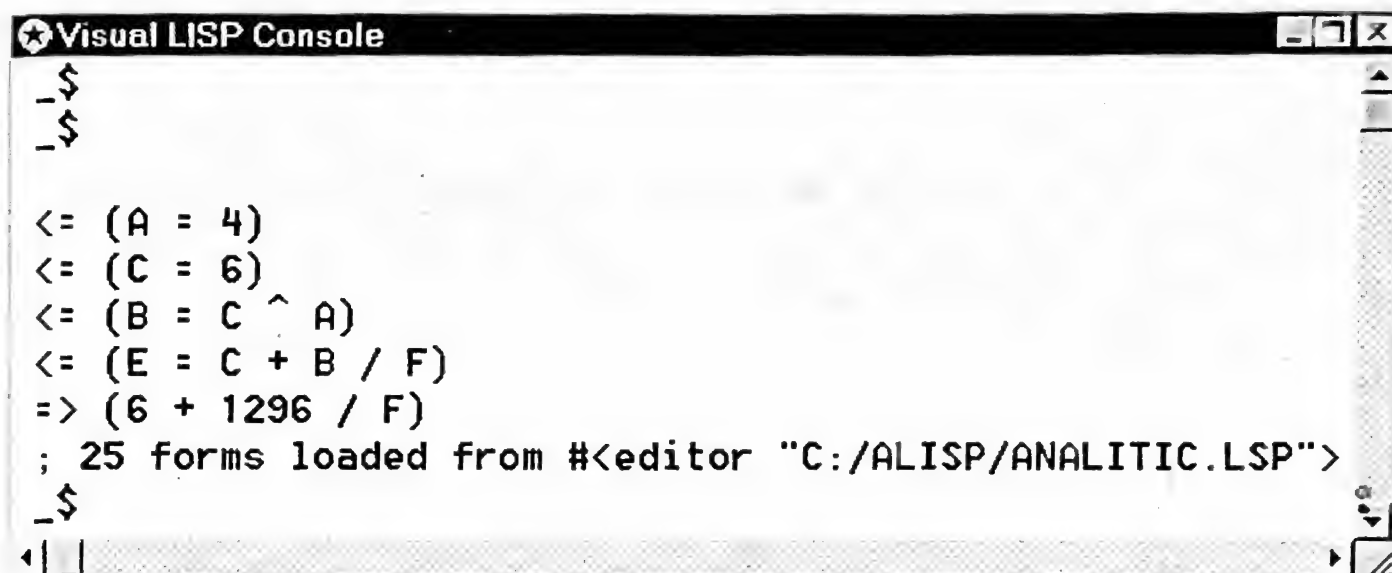
```

Результат вычисления записывается в файл ANALITIC.REZ.

При использовании интегрированной среды Visual LISP после записи функции в файл ANALITIC.LSP без ошибок в окне текстового редактора и запуска ее в окне консоли получим следующий результат (рис. 4.1).

4.1.5. Разработка последовательности действий и программы вычисления производных алгебраических функций

Напомним, что алгебраическая функция – это алгебраическое выражение, в котором некоторые из букв (или все) переменные. Ограничимся исследованием алгебраического выражения с одной переменной, то есть алгебраической функции с одной переменной. Для упрощения изложения используем четыре основные операции – действия: сложение, вычитание, умножение и деление.



```

Visual LISP Console
- $
- $
<= (A = 4)
<= (C = 6)
<= (B = C ^ A)
<= (E = C + B / F)
=> (6 + 1296 / F)
; 25 forms loaded from #<editor "C:/ALISP/ANALITIC.LSP">
- $

```

Рис. 4.1. Окно консоли Visual LISP с результатами преобразований алгебраического выражения

В основу аналитического вычисления производных алгебраических выражений с одной переменной положим хорошо известные основные правила дифференцирования:

$$\begin{aligned}
 D(C * U) &= C * DU ; \\
 D(U + V) &= DU + DV ; \\
 D(U - V) &= DU - DV ; \\
 D(U * V) &= DU * V + U * DV ; \\
 D(U/V) &= (DU * V - U * DV) / (V * V) ,
 \end{aligned}$$

где C – константа;

U, V – алгебраические функции одной переменной X :

$U = F(X); V = K(X)$.

Для вычисления производной алгебраической функции с одной переменной необходимо определить вид операции алгебраической функции, а затем в соответствии с видом операции выполнить дифференцирование. При разработке программы целесообразно использовать рекурсию – процесс определения функции через саму себя.

Разработаем функцию (**DIF..**), которая аналитически вычисляет производную от алгебраического выражения с одной переменной. Функция (**DIF..**) имеет два аргумента: первый – алгебраическое выражение, представленное в префиксной форме; второй – переменная, согласно которой будет проводиться вычисление производной (дифференцирование). Функция (**DIF..**) будет выглядеть так:

```

; *****
; (DIF Y X) - функция дифференцирования алгебраического выражения
; Y в префиксной форме записи. Y представляется в виде бинарного
; списка любого уровня вложенности, например, '(* X (* X X))', т.е.

```

```

; список каждого уровня имеет только два элемента в списке.
; X - переменная, по которой берется производная
; *****
(DEFUN DIF (Y X)
  (TEXTSCR) ; переход на текстовый экран
  (COND
    ((ATOM Y) (IF (EQ Y X) 1 0))
    ((EQ (NTH 0 Y) '+)
      (LIST '+ (DIF (NTH 1 Y) X) (DIF (NTH 2 Y) X)))
    ((EQ (NTH 0 Y) '-')
      (LIST '- (DIF (NTH 1 Y) X) (DIF (NTH 2 Y) X)))
    ((EQ (NTH 0 Y) '*')
      (LIST '+ (LIST '* (DIF (NTH 1 Y) X) (NTH 2 Y))
        (LIST '* (NTH 1 Y) (DIF (NTH 2 Y) X))))
    ((EQ (NTH 0 Y) '/')
      (LIST '/ (LIST '- (LIST '* (DIF (NTH 1 Y) X)
        (NTH 2 Y)) (LIST '* (NTH 1 Y) (DIF (NTH 2 Y) X)))
        (LIST '* (NTH 1 Y) (NTH 1 Y)))))
  )
  (DIF '(* X (* X X)) 'X) ; вызов функции
  (- (* 1 (* X X)) (* X (+ (* 1 X) (* X 1)))) ; результат

```

4.2. Обработка статистических данных

Рассмотрим основные этапы создания системы обработки статистических данных.

4.2.1. Постановка задачи

Требуется разработать программу (функцию) для автоматизации обработки экспериментально-статистических данных. В качестве метода обработки этих данных используется метод наименьших квадратов. Программа должна определить такие параметры уравнения регрессии, которые обеспечат наилучшее расположение линии регрессии среди множества точек поля корреляции. Необходимо построить поле корреляции, изобразить на нем теоретическую линию уравнения регрессии и само уравнение регрессии. Для решения задачи используется язык AutoLISP.

4.2.2. Выявление основных особенностей, взаимосвязей и количественных закономерностей

Ограничимся в данной задаче определением параметров уравнения регрессии линейного вида

$$Y = A_0 + A_1 * X,$$

где A_0 – свободный член уравнения регрессии;

A_1 – коэффициент уравнения регрессии.

Для определения свободного члена A_0 и коэффициента уравнения регрессии A_1 используем общеизвестные формулы:

$$A_0 = \frac{\sum_{i=1}^N Y_i \sum_{i=1}^N X_i^2 - \sum_{i=1}^N Y_i X_i \sum_{i=1}^N X_i}{N \sum_{i=1}^N X_i^2 - \sum_{i=1}^N X_i \sum_{i=1}^N X_i}; \quad A_1 = \frac{N \sum_{i=1}^N Y_i X_i - \sum_{i=1}^N X_i \sum_{i=1}^N Y_i}{N \sum_{i=1}^N X_i^2 - \sum_{i=1}^N X_i \sum_{i=1}^N X_i}$$

где N – число пар исходных экспериментально-статистических данных;

X_i, Y_i – значения 1-го факторного и результативного признаков.

Теснота связи между факторным признаком X_i и результативным признаком Y_i оценивается коэффициентом корреляции R , который определяется по формуле

$$R = \frac{N \sum_{i=1}^N Y_i X_i - \sum_{i=1}^N X_i \sum_{i=1}^N Y_i}{\sqrt{\left(N \sum_{i=1}^N X_i^2 - \sum_{i=1}^N X_i \sum_{i=1}^N X_i \right) \left(N \sum_{i=1}^N Y_i^2 - \sum_{i=1}^N Y_i \sum_{i=1}^N Y_i \right)}}$$

Изображение поля корреляции и всех результатов расчета необходимо произвести с максимально возможным масштабом увеличения. Уравнение регрессии и значение коэффициента корреляции следует изобразить над теоретической линией уравнения регрессии с учетом наклона линии. Для решения задачи требуется максимально использовать встроенные функции языка AutoLISP.

4.2.3. Разработка последовательности действий обработки экспериментально-статистических данных

Обработка экспериментально-статистических данных включает следующие основные этапы:

- *определение исходных сумм:*

$$SX = \sum_{i=1}^N X_i; \quad SY = \sum_{i=1}^N Y_i; \quad SXX = \sum_{i=1}^N X_i^2; \quad SXY = \sum_{i=1}^N X_i Y_i; \quad SY Y = \sum_{i=1}^N Y_i^2$$

- *определение параметров уравнения регрессии A_0, A_1 и коэффициента корреляции R .*

Формулы для определения A_0, A_1, R выглядят несколько громоздко. При этом отдельные их части повторяются в других формулах. Для упрощения расчета введем промежуточные формулы.

$$A = N \sum_{i=1}^N X_i^2 - \sum_{i=1}^N X_i \sum_{i=1}^N X_i$$

$$B = N \sum_{i=1}^N Y_i^2 - \sum_{i=1}^N Y_i \sum_{i=1}^N Y_i$$

$$C = N \sum_{i=1}^N Y_i X_i - \sum_{i=1}^N X_i \sum_{i=1}^N Y_i$$

В таком случае искомые параметры можно будет определить следующим образом:

$$A_0 = \left(\sum_{i=1}^N Y_i \sum_{i=1}^N X_i^2 - \sum_{i=1}^N Y_i X_i \sum_{i=1}^N X_i \right) / A$$

$$A_1 = C/A$$

$$R = C / \sqrt{A \times B}$$

- *изображение поля корреляции.* Необходимо предварительно определить максимальные значения факторного и результативного признаков, размеры осей и местоположение начала координат, а затем оформить список точек для построения поля корреляции, то есть наглядного представления исходных данных на графике.

Размеры осей: $OSX = 1,2 X_{MAX}$; $OSY = 1,2 Y_{MAX}$

Начало координат: $X_0 = 0,1 X_{MAX}$; $Y_0 = 0,1 Y_{MAX}$

- *изображение теоретической линии регрессии в поле корреляции и самого уравнения вдоль линии регрессии.* Для этого необходимо найти начальную и конечную точки для прорисовки теоретической линии уравнения регрессии. Начальная точка может быть определена из уравнения регрессии:

$$Y = A_0 + A_1 * X \text{ при } X = 0, \text{ а конечная при } X = X_{MAX}$$

Зная значения ординат для начальной ТН и конечной ТК точек, можно определить их координаты:

$$(X_0, Y_0 + A_0) \text{ и } (X_{MAX} + X_0, Y_0 + A_0 + A_1 * X_{MAX})$$

Начальную точку ТТ для изображения самого уравнения регрессии примем равной $(X_0, Y_0 + 1,1 A_0)$;

- *формирование надписей вдоль осей X и Y.* Для этого необходимо определить начальные точки надписей, а сами надписи передаются в момент вызова функции.

4.2.4. Разработка фрагментов программы для обработки экспериментально-статистических данных

Разработку функции для обработки статистических данных (MNKV...) на языке AutoLISP начнем с *определения исходных сумм*. При этом будем использовать встроенные функции (APPLY...) и (MAPCAR...), которые часто называют функционалами. *Функционал* – это функция, имеющая функциональный аргумент. Под *функциональным аргументом* понимается аргумент, значением которого является функция.

Функция (APPLY...) – это функционал, который использует для обработки списков другие функции. Данную функцию часто называют применяющим, аппликативным функционалом, поскольку она дает возможность работать со списками с наибольшей эффективностью. Функция (APPLY...) является функцией двух аргументов.

(APPLY <'имя функции> <'(список аргументов)>) – применение заданной функции к списку аргументов.

Например, сумму X_i в нашей задаче можно записать следующим образом:

```
(SETQ SX (APPLY '+ LX))
```

где LX – список значений факторного признака ($X_1 X_2 \dots$).

Для определения сумм $\sum_{i=1}^N X_i^2$, $\sum_{i=1}^N X_i Y_i$, $\sum_{i=1}^N Y_i^2$

необходимо составить соответствующие списки LXX, LXY, LYY.

Это легко сделать с помощью функции (MAPCAR...).

(MAPCAR <'имя функции> <'(список аргументов)>...) – выполнение заданной функции над первыми элементами списков, затем вторыми и т. д.

```
(SETQ LXX (MAPCAR '* LX LX)
      LXY (MAPCAR '* LX LY)
      LYY (MAPCAR '* LY LY)
)
```

где LY – список значений результативного признака ($Y_1 Y_2 \dots$).

Для проведения последующих расчетов необходимо знать число пар исходных данных. Для этого используется встроенная функция (LENGTH...), имеющая один аргумент – список.

(LENGTH <список>) – определение длины списка.

Данная функция определяет длину списка, а следовательно, число пар исходных данных. Эта часть программы (функция) будет выглядеть так:

```
(SETQ SX (APPLY '+ LX) SY (APPLY '+ LY)
      LXX (MAPCAR '* LX LX) LXY (MAPCAR '* LX LY)
```

```

LYY (MAPCAR '* LY LY) SXX (APPLY '+ LXX)
SXY (APPLY '+ LXY) SY (APPLY '+ LYY)
PI05 (* PI 0.5) N (LENGTH LX)

```

Для определения параметров уравнения регрессии A_0 , A_1 и коэффициента корреляции R используем встроенные функции сложения (+...), умножения (*...) и деления (/...):

```

(SETQ  A (- (* N SXX) (* SX SX))
        B (- (* N SY) (* SY SY))
        A0 (/ (- (* SY SXX) (* SXY SX)) A)
        C (- (* N SXY) (* SX SY))
        A1 (/ C A)
        R (/ C (SQRT (* A B)))
)

```

Формирование изображения поля корреляции включает определение размеров поля корреляции и собственно вывод графического изображения. Размеры поля корреляции представляются максимальными значениями факторного и результативного признаков, которые вычисляются с использованием функций (APPLY...) и (MAX...):

```

(SETQ  MAXX (APPLY 'MAX LX)  MAXY (APPLY 'MAX LY)
        K (/ 220 MAXX)      LX (MAPCAR 'UMN LX)
        KX K                K (/ 170 MAXY)
        KXY (/ K KX)        LY (MAPCAR 'UMN LY)
        LT (MAPCAR 'CONS LX LY) TO (LIST 20 20)
        OX (POLAR TO 0 220)  OY (POLAR TO PI05 170)
)

```

С помощью команд графического редактора AutoCAD "ZOOM", "ERASE" и "PLINE" определим масштаб отображения поля корреляции, удалим все объекты внутри заданного окна и выведем оси координат (рис. 4.2).

```

(COMMAND "ZOOM" "W" "0,0" "250,200"
  "ERASE" "W" "0,0" "250,200" ""
  "PLINE" OX "W" 0.1 0.1 TO OY "")

```

Формирование изображения поля корреляции осуществляется с помощью функции (FOREACH T LT...), которая из списка LT последовательно выделяет точку T, и команды "POINT", которая рисует точку в поле корреляции:

```

(FOREACH T LT
  (SETQ T1 (POLAR TO 0 (CAR T))
        T2 (POLAR T1 PI05 (CDR T))
  )
  (COMMAND "POINT" T2)
)

```

Формирование изображения линии уравнения регрессии можно выполнить следующим образом:

```
(SETQ TR (POLAR TO PI05 (* K A0))
      YT (+ (* K A0) (* A1 KXY 220))
      T1 (POLAR TO 0 220)
      T2 (POLAR T1 PI05 YT)
)
(COMMAND "PLINE" TR "W" 0.2 0.2 T2 "")
```

Для изображения над линией самого уравнения регрессии используем команду **"TEXT"**, которой, в свою очередь, требуется начальная точка текста (ТТ), угол наклона текста UGOL и сам текст. Начальную точку текста определим с помощью функции (LIST...):

```
(LIST XO (+ YO (* A0 1.1)))
```

Для определения угла наклона уравнения регрессии используем встроенную функцию (ANGLE T1 T2), которая определяет в радианах угол линии, проходящей через точки T1 и T2 относительно горизонтальной оси против часовой стрелки. Однако угол наклона текста в команде **"TEXT"** должен быть задан в градусах. Следовательно, радианы надо перевести в градусы, и это может быть сделано с помощью комбинации функций:

```
(/ (* 180 (ANGLE TR T2)) PI)
```

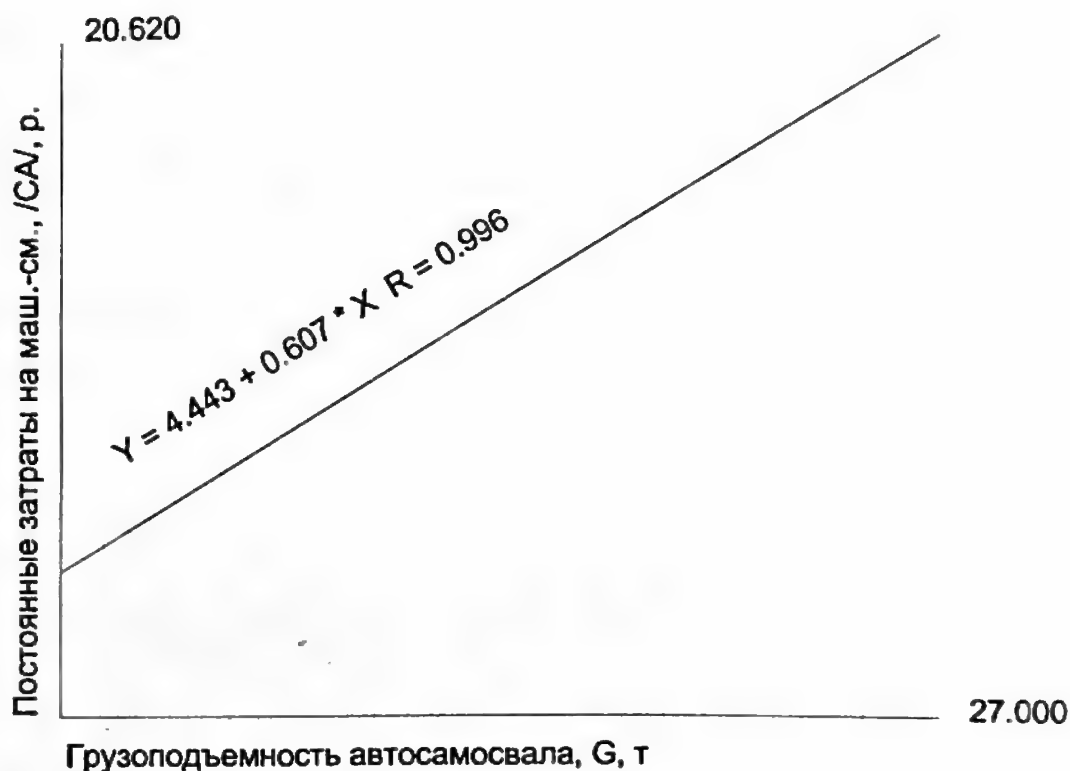


Рис. 4.2. Изображение поля корреляции, осей координат и уравнения регрессии

Текст можно сформировать, используя функцию (**STRCAT...**).

(**STRCAT** <"текст1"> <"текст2">...)

Эта функция сцепляет строковые константы (тексты, взятые в двойные кавычки) в единый текст – текстовую константу. Присвоим ей имя **URAV**.

Допустим, что нам необходимо иметь запись уравнения над линией регрессии в следующем виде:

$$Y = 0.605 + 4.52 * X, R = 0.9568$$

Эта строка содержит изменяемые значения, поэтому представить ее в виде одной строковой константы можно так:

```
(STRCAT " Y = " (RTOS A0 2 3) " + " (RTOS A1 2 2) " * X "
" R = " (RTOS R 2 4))
```

Здесь используется функция (**RTOS...**).

(**RTOS** <действительное число> <формат числа> <точность>)

Данная функция преобразует действительное число в заданный формат числа с указанной точностью, а затем в строковую константу. Если формат числа равен 1, а точность 2, то это научный формат представления чисел с точностью до двух знаков после точки. Если формат числа равен 2, а точность 3, то это десятичный формат представления действительного числа с точностью до 3 знаков после точки.

Таким образом, определение угла наклона отображения уравнения регрессии **UGOL** и формирование текста уравнения **URAV** можно представить в следующем виде:

```
(SETQ URAV (STRCAT " Y = " (RTOS A0 2 3)
" + " (RTOS A1 2 3) " * X "
" R = " (RTOS R 2 3))
UGOL (/ (* 180 (ANGLE TR T2)) PI)
TT (LIST 20 (+ 40 (* A0 K)))
)
```

Для задания стиля текста и его вывода используем команды графического редактора AutoCAD **"STYLE"** и **"TEXT"**:

```
(COMMAND "STYLE" "STANDARD" "TXT" "4" "1" "" "" "" ""
"TEXT" TT UGOL URAV
"TEXT" OX 0 (RTOS MAXX 2 3)
"TEXT" OY PI05 (RTOS MAXY 2 3))
```

Напомним, что команда **"STYLE"** имеет следующие параметры:

- имя текстового стиля – по умолчанию **"STANDARD"**;
- имя файла шрифта – по умолчанию **"TXT"**;
- высота шрифта – 4;
- степень сжатия/расширения – 1;

- угол наклона шрифта по умолчанию ноль градусов;
- порядок записи символов по умолчанию слева направо;
- вид записи по умолчанию в неперевернутом виде;
- ориентация записи текста горизонтальная или вертикальная (по умолчанию горизонтальная).

Для формирования надписей вдоль осей X и Y определим начальные точки надписей с помощью прямого их указания, а сами надписи поместим в переменные TX, TY:

```
(COMMAND "TEXT" "20,10" 0 TX
        "TEXT" "10,20" 90 TY)
```

Ниже представлена функция (MNKV...) расчета параметров уравнения регрессии первого порядка и изображения поля корреляции и теоретического уравнения регрессии.

4.2.5. Программа для обработки экспериментально-статистических данных

```
; *****
; (MNKV LX LY TX TY) - функция расчета параметров уравнения
; регрессии первого порядка и изображения поля
; корреляции и теоретического уравнения регрессии
; А р г у м е н т ы   ф у н к ц и и:
; LX - список значений факторного признака ' (X1 X2...)
; LY - список значений результативного признака ' (Y1 Y2...)
; TX - поясняющий текст вдоль оси X
; TY - поясняющий текст вдоль оси Y
; Пример вызова функции (MNKV LX LY) -
; (MNKV ' (2.25 4.5 7.0 10.0 12.0 27.0)
; ' (5.33 6.8 9.32 11.07 11.62 20.62)
; "Грузоподъемность автосамосвала, G, т "
; "Постоянные затраты на маш.-см., /CA/,р.")
; *****
(DEFUN UMN (X)
  (* X K)
)

(DEFUN MNKV (LX LY TX TY)
  (SETVAR "CMDECHO" 0)
  (SETQ SX (APPLY '+ LX) '          SY (APPLY '+ LY)
        LXX (MAPCAR '* LX LX)      LXY (MAPCAR '* LX LY)
        LYY (MAPCAR '* LY LY)      SXX (APPLY '+ LXX)
        SXY (APPLY '+ LXY)         SYX (APPLY '+ LYY)
        PI05 (* PI 0.5)           N (LENGTH LX)
  )
)
```

```

(SETQ A (- (* N SXX) (* SX SX))
      B (- (* N SY) (* SY SY))
      A0 (/ (- (* SY SXX) (* SXY SX)) A)
      C (- (* N SXY) (* SX SY))
      A1 (/ C A)
      R (/ C (SQRT (* A B)))
)
(SETQ MAXX (APPLY 'MAX LX)      MAXY (APPLY 'MAX LY)
      K (/ 220 MAXX)           LX (MAPCAR 'UMN LX)
      KX K                      K (/ 170 MAXY)
      KXY (/ K KX)             LY (MAPCAR 'UMN LY)
      LT (MAPCAR 'CONS LX LY)  TO (LIST 20 20)
      OX (POLAR TO 0 220)      OY (POLAR TO PI05 170)
)
(COMMAND "ZOOM" "W" "0,0" "250,200"
      "ERASE" "W" "0,0" "250,200" ""
      "PLINE" OX "W" 0.1 0.1 TO OY "")
(Foreach TT LT
  (SETQ T1 (POLAR TO 0 (CAR TT))
        T2 (POLAR T1 PI05 (CDR TT)))
  (COMMAND "POINT" T2)
)
(SETQ TR (POLAR TO PI05 (* K A0))
      YT (+ (* K A0) (* A1 KXY 220))
      T1 (POLAR TO 0 220)
      T2 (POLAR T1 PI05 YT)
)
(COMMAND "PLINE" TR "W" 0.2 0.2 T2 "")
(SETQ URAV (STRCAT " Y = " (RTOS A0 2 3)
                  " + " (RTOS A1 2 3) " * X "
                  " R = " (RTOS R 2 3))
      UGOL (/ (* 180 (ANGLE TR T2)) PI)
      TT (LIST 20 (+ 40 (* A0 K))))
(COMMAND "STYLE" "" "" "4" "1" "" "" "" ""
      "TEXT" TT UGOL URAV
      "TEXT" OX 0 (RTOS MAXX 2 3)
      "TEXT" OY PI05 (RTOS MAXY 2 3)
)
(COMMAND "TEXT" "20,10" 0 TX
      "TEXT" "10,20" 90 TY)
)
(MNKV '(2.25 4.5 7.0 10.0 12.0 27.0) ; вызов функции
      '(5.33 6.8 9.32 11.07 11.62 20.62)

```

"Грузоподъемность автосамосвала, G, т"

"Постоянные затраты на маш.-см., /CA/, р.")

После вызова функции (MNKV...) на экране появится графическое изображение поля корреляции, теоретической линии и уравнения регрессии с рассчитанными параметрами (рис. 4.3).

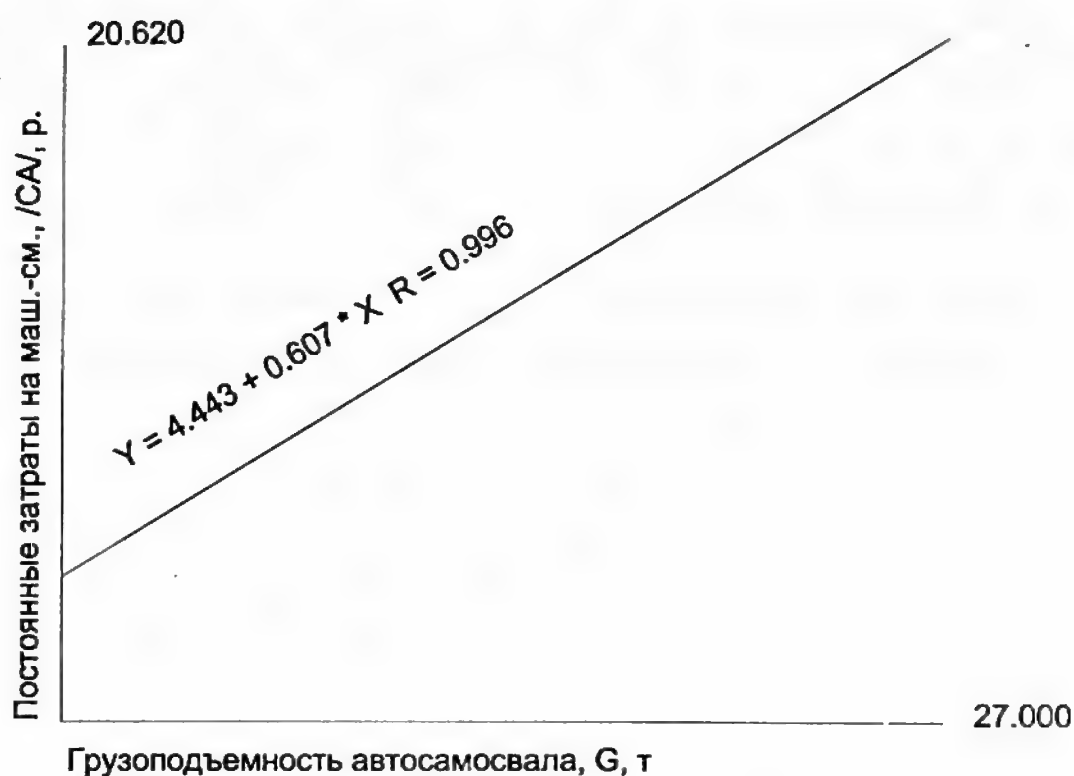


Рис. 4.3. Результат выполнения функции (MNKV...)

4.3. Разработка простейшей реляционной базы данных

Базы данных являются исходным этапом выработки и принятия различного рода решений. Рассмотрим основные этапы разработки простейшей реляционной базы данных, в основе которой лежит табличная форма представления данных.

4.3.1. Постановка задачи

Требуется создать простейшую реляционную базу данных на основе использования встроенных функций языка программирования AutoLISP. Разрабатываемая база данных должна обеспечить возможность поиска необходимой информации, обновления и вывода ее на экран. В качестве исходной информации возьмем технические параметры строительных универсальных экскаваторов, представленные в виде таблицы (см. табл. 4.1).

Таблица 4.1

Параметры	Обозн.	ЭО-3122	ЭО-43216	ЭО-5124	ЭО-6123
Эксплуатационная масса, т	G	14,30	19,50	39,00	67,50
Вместимость ковша	Q	0,63	0,80	1,60	2,50
Наиб. глубина копания, м	Нк	4,80	5,50	6,50	7,20
Наиб. радиус копания, м	Rк	7,75	8,85	10,00	11,60
Наиб. высота выгрузки, м	Нв	4,50	5,50	5,50	5,80
Мощность двигателя, кВт	N	55,10	73,60	121,00	150,00
Мин. длительность цикла, с	ТС	16,30	19,60	25,00	28,00

4.3.2. Выявление основных особенностей, взаимосвязей и количественных закономерностей

Данные, собранные в таблице, можно представить:

- в виде совокупности списка имен параметров;
- ' ("G" "Q" "HK" "RK" "HV" "N" "TC")
- в виде списков значений параметров для каждого экскаватора. Например, для экскаватора ЭО-5124 этот список будет выглядеть так:
- ' (39.0 1.6 6.5 10.0 5.5 121 25.0)

Используем следующие переменные:

P – имя параметра;

PZ – значение параметра;

LP – список имен параметров;

LE – список '(P PZ), состоящий из двух элементов: имени параметра P и его значения PZ;

LA – ассоциативный список, состоящий из списков LE.

Представлять табличные данные в виде списков удобно тем, что AutoLISP имеет полный набор функций создания, поиска информации в ассоциативных списках по ключу (имени параметра), обновления данных в списке, а также другие функции. Ассоциативный список LA является хранилищем данных одного столбца таблицы.

4.3.3. Разработка последовательности действий и программы создания базы данных

Создание простейшей реляционной (табличной) базы данных включает следующие этапы:

- формирование списка имен параметров (ключей);
- ввод для каждого параметра его значения;

- формирование пар (списков), состоящих из имен параметра (ключа) и его значения;
- формирование списка пар – ассоциативного списка.

Поиск значения заданного параметра включает:

- поиск пары, содержащей заданный параметр (ключ);
- выделение из пары значения заданного параметра.

Обновление данных включает следующие этапы:

- поиск в ассоциативном списке пар обновляемой пары по заданному параметру (ключу);
- формирование новой пары;
- замену обновляемой пары в списке пар на новую пару.

Процесс создания простейшей базы данных, поиск нужного значения параметра и обновление проведем в диалоговом режиме.

Для этого разработаем три функции: функцию создания базы данных (**SBD...**), функцию поиска значения заданного параметра (**PBD**) и функцию для обновления данных (**OBD**).

Сформируем список имен нужных параметров – '(LP), например:

```
(SETQ LP (LIST "G" "Q" "HK" "RK" "HV" ))
```

Для ввода параметров используем функцию (**GETSTRING...**), которая предназначена для ввода текстовой константы:

```
(SETQ PZ (GETSTRING))
```

Пары (списки), состоящие из имен параметров (ключа) и значения параметра, создадим с помощью функции (**LIST...**) – функции формирования списка:

```
(SETQ LE (LIST P PZ))
```

Создадим ассоциативный список с помощью (**CONS...**) – функции, которая формирует список путем добавления в его начало нового элемента:

```
(SETQ LA (CONS LE LA))
```

Ниже представлена функция (**SBD...**) для создания простейшей базы данных в диалоговом режиме.

4.3.4. Программа создания базы данных в диалоговом режиме

```
; *****
; (SBD LP) - функция создания простейшей базы данных
; в диалоговом режиме
; LP - список вводимых в базу параметров
; *****
```

```

(DEFUN SBD (LP)
  (SETQ LA '()) ; ассоциативный список
  (TEXTSCR)
  (FOREACH P LP ; цикл ввода значений параметров
    (PRINC (STRCAT "\n Введите значение параметра " P " = "))
    (SETQ PZ (GETSTRING) ; значение параметра
      LE (LIST P PZ) ; пары параметр-значение
      LA (CONS LE LA))) ; список пар параметр-значение
  (SETQ LA (REVERSE LA)) ; обращение списка пар
  (SBD '("G" "Q" "HK" "RK" "HV")) ; вызов функции (SBD...)

```

В диалоговом режиме запрашиваются значения следующих параметров:

Введите значение параметра G = 14.3
 Введите значение параметра Q = 0.63
 Введите значение параметра HK = 4.8
 Введите значение параметра RK = 7.75
 Введите значение параметра HV = 4.5

Параметры и их значения будут представлены в виде ассоциативного списка LA:

```
(( "G" "14.3" ) ( "Q" "0.63" ) ( "HK" "4.8" ) ( "RK" "7.75" ) ( "HV" "4.5" ))
```

Поскольку эта информация относится только к экскаватору ЭО-3122, запомним ее в переменной с именем EO-3122.

```
(SETQ EO-3122 LA)
```

Для сохранения этих данных они должны быть записаны в файл хранения данных. Перед использованием базы данных этот файл должен быть загружен.

Функция (SETQ LA '()) задает пустое начальное значение списка пар ассоциативного списка LA.

Выражение (FOREACH P LP...) последовательно в цикле присваивает переменной P очередное имя параметра из списка. Далее для этого параметра выполняются функция (PRINC...) вывода на экран выражения, сформированного функцией (STRCAT...), функция (LIST...) образования пары параметр-значение, функция (CONS...) формирования списка пар параметр-значение.

Функция (REVERSE LA) изменяет порядок следования элементов на обратный.

Функция поиска значения заданного параметра (PBD) должна начинаться с ввода искомого параметра:

```
(SETQ P (GETSTRING "\n Введите параметр (или Enter) "))
```

Отказаться от поиска можно, если нажать клавишу **Enter**. Далее с помощью встроенной функции (ASSOC...) выделим пару по заданному па-

параметру (ключу). С помощью функции (CADR...) из пары выделим искомое значение параметра.

Функции (PRINC...) и (STRCAT...) обеспечивают вывод найденного значения параметра, например, TC = 25.0.

Функция поиска значения нужного параметра (PBD) будет выглядеть следующим образом.

4.3.5. Программа поиска данных в базе данных в диалоговом режиме

```
; *****
; (PBD LA) - функция поиска значений нужного
; параметра в базе данных в диалоговом режиме
; LA - ассоциативный список пар (параметр значение)
; перед использованием функции (PBD) необходимо сфор-
; мировать соответствующую базу данных функцией (SBD)
; *****
(DEFUN PBD (LA)
  (SETQ P (GETSTRING "\n Введите искомый параметр (или Enter) "))
  (IF (/= (ASCII P) 10)
    (PROGN (SETQ LE (ASSOC P LA)
              PZ (CADR LE))
      (PRINC (STRCAT P " = " PZ))))
  )
```

Допустим, функция (SBD...) сформировала ассоциативный список под именем LA следующего вида:

```
(( "G" "39.0") ("Q" "1.6") ("HK" "6.5") ("RK" "10.0") ("HV" "5.5"))
```

Поскольку эта информация относится только к экскаватору ЭО-5124, запомним ее в переменной с именем EO-5124.

```
(SETQ EO-5124 LA)
```

Для сохранения этих данных они должны быть записаны в файл хранения данных. Перед поиском данных этот файл должен быть загружен.

Если в ответ на запрос функции (PBD EO-5124) ввести следующее значение:

Введите искомый параметр (или Enter) Q

получим результат

Q = 1.6

Функция обновления данных выполняет те же действия, что и функция поиска, а также действия, связанные с вводом нового значения параметра и заменой старого значения на новое.

Ниже представлена функция для обновления данных (OBD).

4.3.6. Программа обновления значений в базе данных в диалоговом режиме

```
; *****
; (OBD LA) - функция обновления значений в базе данных
; в диалоговом режиме
; LA - ассоциативный список пар (параметр значение)
; предварительно должна быть создана БД функцией (SBD)
; *****
(DEFUN OBD (LA)
  (SETQ P (GETSTRING "\n Введите обновляемый параметр "))
  (IF (/= (ASCII P) 10)
    (PROGN
      (SETQ LE (ASSOC P LA)
        PZ (CADR LE))
      (PRINC (STRCAT P " = " PZ))
      (SETQ PZN (GETSTRING "\n Введите новое значение "))
      (IF (/= (ASCII PZN) 10)
        (SETQ PZN (LIST P PZN)
          LA (SUBST PZN LE LA))))))
)
```

Допустим, что функция (**SBD...**) сформировала ассоциативный список под именем LA вида:

```
(( "G" "67.5" ) ( "Q" "2.5" ) ( "HK" "7.2" ) ( "RK" "11.6" ) ( "HV" "5.8" ))
```

Поскольку эта информация относится только к экскаватору ЭО-6123, запомним ее в переменной с именем EO-6123.

```
(SETQ EO-6123 LA)
```

Для сохранения этих данных их можно записать в файл хранения данных. Перед обновлением данных этот файл должен быть загружен.

В ответ на запрос функции (**OBD EO-6123**) необходимо ввести обновляемый параметр и новое значение.

Введите обновляемый параметр RK

Введите новое значение 11.8

Тогда ассоциативный список под именем **EO-6123** будет уже выглядеть так:

```
(( "G" "67.5" ) ( "Q" "2.5" ) ( "HK" "7.2" ) ( "RK" "11.8" ) ( "HV" "5.8" ))
```

СОЗДАНИЕ ОБЪЕКТНО-ОРИЕНТИРОВАННЫХ СИСТЕМ

Расчет электрических и технических систем	192
Сетевое планирование и управление проектами	215
Разработка экспертной системы	233

В этой главе приведены примеры создания различных объектно-ориентированных систем (изображения и расчета электрических и механических подсистем, систем сетевого планирования и управления), а также простейшей экспертной системы.

5.1. Расчет электрических и технических систем

Рассмотрим создание таких распространенных подсистем, как формирование графического изображения и расчет электрических схем переменного тока, расчет собственных частот механических систем.

5.1.1. Постановка задачи

Имеются электрические схемы переменного тока, состоящие из сопротивлений, индуктивностей и конденсаторов. Требуется разработать объектно-ориентированную подсистему для формирования графических изображений электрических схем любой сложности с прорисовкой как самих элементов, так и их параметров.

5.1.2. Выявление основных особенностей, взаимосвязей и количественных закономерностей

Элементы электрической схемы могут быть соединены последовательно либо параллельно.

Для создания графического изображения схемы могут понадобиться горизонтальные и вертикальные проводники, соединяющие те или иные элементы схемы. Вертикальные проводники могут быть направлены вверх или вниз, горизонтальные – вправо или влево.

Каждый базовый элемент электрической схемы можно представить в виде некоторой функции. Для формирования графического изображения сопротивления необходимо создать функцию (RP), для индуктивности – (LP), для конденсатора – (CP), для вертикального проводника, направленного вверх, – (VP+), вниз – (VP–), для горизонтального проводника, направленного вправо, – (GP+), влево – (GP–).

5.1.3. Разработка последовательности действий и комплекса функций параметрического изображения всех элементов схемы

Прежде создадим функции для формирования графических изображений базовых элементов схемы:

```
(DEFUN RP () ; функция создания изображения резистора
(COMMAND "PLINE" TB "W" "0.01" "0.01" "@2,0" "@0,1" "@6,0"
"@0,-1" "@2,0" "@-2,0" "@0,-1" "@-6,0" "@0,1" ""
"TEXT" (LIST (+ X 2) (+ Y 1.5)) "0.7" "0" "R"
```

```

"TEXT" (LIST (+ X 3) (+ Y 1.5)) "0.7" "0" NOM
"TEXT" (LIST (+ X 5) (+ Y 1.5)) "0.7" "0" NUM)
)
(DEFUN LP () ; функция изображения катушки индуктивности
(COMMAND "PLINE" TB "W" "0.01" "0.01" "@2,0" "@1,1" "@1,-2"
"@1,2" "@1,-2" "@1,2" "@1,-1" "@2,0" ""
"TEXT" (LIST (+ X 2) (+ Y 1.5)) "0.7" "0" "L"
"TEXT" (LIST (+ X 3) (+ Y 1.5)) "0.7" "0" NOM
"TEXT" (LIST (+ X 5) (+ Y 1.5)) "0.7" "0" NUM)
)
(DEFUN CP () ; функция создания изображения конденсатора
(COMMAND
"PLINE" TB "W" "0.01" "0.01" "@4,0" "@0,1" "@0,-2" ""
"PLINE" "@2,0" "W" "0.01" "0.01" "@0,2" "@0,-1" "@4,0" ""
"TEXT" (LIST (+ X 2) (+ Y 1.5)) "0.7" "0" "C"
"TEXT" (LIST (+ X 3) (+ Y 1.5)) "0.7" "0" NOM
"TEXT" (LIST (+ X 5) (+ Y 1.5)) "0.7" "0" NUM)
)

```

В функциях используются переменные TB, NOM и NUM, которые определяют базовую точку элемента, номер и величину сопротивления элемента соответственно. Эти переменные должны быть определены до использования функций (RP), (LP) и (CP), например, так:

```
(SETQ TB (LIST 5 7) NOM "3" NUM "1000")
```

Ниже на рис. 5.1 представлены примеры создания в среде AutoCAD графических изображений резистора, катушки индуктивности и конденсатора.



Рис. 5.1. Примеры графического изображения элементов электрической схемы

Создадим функции формирования графического изображения проводников.

```

(DEFUN VP+ () ; функция изображения вертикального проводника вверх
(COMMAND "PLINE" TB "W" "0.01" "0.01" "@0,10" ""))
(DEFUN VP- () ; функция изображения вертикального проводника вниз
(COMMAND "PLINE" TB "W" "0.01" "0.01" "@0,-10" ""))
(DEFUN GP+ () ; функция изображения горизонтального проводника вправо
(COMMAND "PLINE" TB "W" "0.01" "0.01" "@10,0" ""))
(DEFUN GP- () ; функция изображения горизонтального проводника влево
(COMMAND "PLINE" TB "W" "0.01" "0.01" "@-10,0" ""))

```


Ввести в диалоговом режиме необходимую исходную информацию можно следующим образом:

```
(SETQ N (GETINT "\n Введите число элементов в цепи N = ")
  I 0)
(REPEAT N ; цикл ввода элементов и их данных
  (SETQ I (+ I 1))
  (SETQ IMY (GETSTRING (STRCAT "\n Введите имя "
    (ITOA I) "-го элемента цепи <R,L,C,VP+,VP-,GP+,GP-> - ")))
  (COND ((EQ IMY "R") (SETQ RAZM " в омах " RAZ " ом"))
    ((EQ IMY "L") (SETQ RAZM " в генри " RAZ " генри"))
    ((EQ IMY "C") (SETQ RAZM " в мкФ " RAZ " мкФ")))
  )
  (COND ((OR (EQ IMY "R") (EQ IMY "L") (EQ IMY "C")))
    (SETQ NOM (ITOA (GETINT "\n Введите номер элемента NOM = ")))
    (SETQ SOPR (STRCAT (GETSTRING (STRCAT "\n Введите "
      "сопротивление элемента цепи " IMY NOM RAZM)) RAZ)))
  (SETQ TB (GETPOINT "\n Введите точку вставки элемента - "))
  (SETQ X (NTH 0 TB)
    Y (NTH 1 TB))
  (COND
    ((EQ IMY "R") (RP)) ; создание изображения резистора
    ((EQ IMY "L") (LP)) ; создание изображения индуктивности
    ((EQ IMY "C") (CP)) ; создание изображения конденсатора
    ((EQ IMY "VP+") (VP+)) ; создание изображения провода вверх
    ((EQ IMY "GP+") (GP+)) ; создание изображения провода вправо
    ((EQ IMY "VP-") (VP-)) ; создание изображения провода вниз
    ((EQ IMY "GP-") (GP-)) ; создание изображения провода влево
  )
)
```

В окончательном варианте комплекс функций для создания графического изображения электрической схемы переменного тока, состоящей из сопротивлений, катушек индуктивностей и конденсаторов, будет выглядеть так, как показано ниже.

5.1.4. Комплекс функций параметрического изображения всех элементов электрической схемы

```
; *****
; Комплекс функций для создания в диалоговом режиме графического
; изображения электрической схемы, состоящей из сопротивлений,
; конденсаторов и катушек индуктивностей
; *****
```

```

(SETVAR "BLIPMODE" 0) ; отключение маркеров
(SETVAR "CMDECHO" 0) ; отключение эха команд
(DEFUN RP () ; функция изображения резистора
  (COMMAND "PLINE" TB "W" "0.01" "0.01" "@2,0" "@0,1" "@6,0"
    "@0,-1" "@2,0" "@-2,0" "@0,-1" "@-6,0" "@0,1" ""
    "TEXT" (LIST (+ X 2) (+ Y 1.5)) "0.7" "0" "R"
    "TEXT" (LIST (+ X 3) (+ Y 1.5)) "0.7" "0" "NOM"
    "TEXT" (LIST (+ X 5) (+ Y 1.5)) "0.7" "0" "SOPR")
)
(DEFUN LP () ; функция изображения катушки индуктивности
  (COMMAND "PLINE" TB "W" "0.01" "0.01" "@2,0" "@1,1" "@1,-2"
    "@1,2" "@1,-2" "@1,2" "@1,-1" "@2,0" ""
    "TEXT" (LIST (+ X 2) (+ Y 1.5)) "0.7" "0" "L"
    "TEXT" (LIST (+ X 3) (+ Y 1.5)) "0.7" "0" "NOM"
    "TEXT" (LIST (+ X 5) (+ Y 1.5)) "0.7" "0" "SOPR")
)
(DEFUN CP () ; функция изображения конденсатора
  (COMMAND
    "PLINE" TB "W" "0.01" "0.01" "@4,0" "@0,1" "@0,-2" ""
    "PLINE" "@2,0" "W" "0.01" "0.01" "@0,2" "@0,-1" "@4,0" ""
    "TEXT" (LIST (+ X 2) (+ Y 1.5)) "0.7" "0" "C"
    "TEXT" (LIST (+ X 3) (+ Y 1.5)) "0.7" "0" "NOM"
    "TEXT" (LIST (+ X 5) (+ Y 1.5)) "0.7" "0" "SOPR")
)
; функция изображения вертикального проводника, направленного вверх
(DEFUN VP+ () (COMMAND "PLINE" TB "W" "0.01" "0.01" "@0,10" ""))
)
; функция изображения вертикального проводника, направленного вниз
(DEFUN VP- ()
  (COMMAND "PLINE" TB "W" "0.01" "0.01" "@0,-10" "")
)
)
; функция изображения горизонтального проводника, направленного вправо
(DEFUN GP+ ()
  (COMMAND "PLINE" TB "W" "0.01" "0.01" "@10,0" "")
)
)
; функция изображения горизонтального проводника, направленного влево
(DEFUN GP- ()
  (COMMAND "PLINE" TB "W" "0.01" "0.01" "@-10,0" "")
)
)
(COMMAND "LIMITS" "0,0" "40, 30")
(COMMAND "ZOOM" "A")
(SETQ N (GETINT "\n Введите число элементов в цепи N = "))
I 0)

```

```

(REPEAT N ; цикл ввода элементов цепи и их данных
  (SETQ I (+ I 1))
  (SETQ IMY (GETSTRING (STRCAT "\n Введите имя "
    (ITOA I) "-го элемента цепи <R,L,C,VP+,VP-,GP+,GP-> - ")))
  (COND ((EQ IMY "R") (SETQ RAZM " в омах " RAZ " ом"))
        ((EQ IMY "L") (SETQ RAZM " в генри " RAZ " генри"))
        ((EQ IMY "C") (SETQ RAZM " в мкФ " RAZ " мкФ")))
  )
  (COND ((OR (EQ IMY "R") (EQ IMY "L") (EQ IMY "C")))
    (SETQ NOM (ITOA (GETINT "\n Введите номер элемента NOM = ")))
    (SETQ SOPR (STRCAT (GETSTRING (STRCAT "\n Введите "
      "сопротивление элемента цепи " IMY NOM RAZM )) RAZ))))
  (SETQ TB (GETPOINT "\n Введите точку вставки элемента - "))
  (SETQ X (NTH 0 TB)
        Y (NTH 1 TB))
  (COND
    ((EQ IMY "R") (RP)) ; создание изображения резистора
    ((EQ IMY "L") (LP)) ; создание изображения индуктивности
    ((EQ IMY "C") (CP)) ; создание изображения конденсатора
    ((EQ IMY "VP+") (VP+)) ; создание изображения провода вверх
    ((EQ IMY "GP+") (GP+)) ; создание изображения провода вправо
    ((EQ IMY "VP-") (VP-)) ; создание изображения провода вниз
    ((EQ IMY "GP-") (GP-))) ; создание изображения провода влево
  )

```

Для формирования в диалоговом режиме графического изображения электрической схемы переменного тока, состоящей из сопротивлений, конденсаторов, катушек индуктивностей и проводников, достаточно обратиться к комплексу функций, созданных выше. Данный комплекс функций целесообразно сохранить в файле, например, ELKTRISD.LSP. Допустим, этот файл находится в папке (каталоге) ALISP на диске C:. После загрузки файла в командной строке AutoCAD

Command: (LOAD "C:/ALISP/ELKTRISD")

будет запрашиваться необходимая информация для создания графического изображения схемы. В качестве примера возьмем схему, изображенную выше. На запросы командной строки AutoCAD введем данные, представленные ниже в правой части запросов. Все запросы и данные по ним будут отражаться в текстовом окне AutoCAD (Shift+F2).

Введите число элементов в цепи N = 12

Введите имя 1-го элемента цепи <R,L,C,VP+,VP-,GP+,GP-> R

Введите номер элемента NOM = 1

Введите сопротивление элемента цепи R1 в омах 1

Введите точку вставки элемента - 0,12

Введите имя 2-го элемента цепи <R,L,C,VP+,VP-,GP+,GP-> VP+
Введите точку вставки элемента - 10,12
Введите имя 3-го элемента цепи <R,L,C,VP+,VP-,GP+,GP-> VP-
Введите точку вставки элемента - 10,12
Введите имя 4-го элемента цепи <R,L,C,VP+,VP-,GP+,GP-> R
Введите номер элемента NOM = 2
Введите сопротивление элемента цепи R2 в омах 100
Введите точку вставки элемента - 10,22
Введите имя 5-го элемента цепи <R,L,C,VP+,VP-,GP+,GP-> L
Введите номер элемента NOM = 1
Введите сопротивление элемента цепи L1 в генри 0.2
Введите точку вставки элемента - 20,22
Введите имя 6-го элемента цепи <R,L,C,VP+,VP-,GP+,GP-> C
Введите номер элемента NOM = 1
Введите сопротивление элемента цепи C1 в мкФ 1000
Введите точку вставки элемента - 10,12
Введите имя 7-го элемента цепи <R,L,C,VP+,VP-,GP+,GP-> R
Введите номер элемента NOM = 3
Введите сопротивление элемента цепи R3 в омах 1000
Введите точку вставки элемента - 10,2
Введите имя 8-го элемента цепи <R,L,C,VP+,VP-,GP+,GP-> GP+
Введите точку вставки элемента - 20,12
Введите имя 9-го элемента цепи <R,L,C,VP+,VP-,GP+,GP-> VP-
Введите точку вставки элемента - 30,22
Введите имя 10-го элемента цепи <R,L,C,VP+,VP-,GP+,GP-> VP+
Введите точку вставки элемента - 30,2
Введите имя 11-го элемента цепи <R,L,C,VP+,VP-,GP+,GP-> GP+
Введите точку вставки элемента - 20,2
Введите имя 12-го элемента цепи <R,L,C,VP+,VP-,GP+,GP-> GP+
Введите точку вставки элемента - 30,12

Фрагмент введенных в диалоговом режиме данных для создания графического изображения электрической схемы представлен в текстовом окне AutoCAD (Shift+F2) (рис. 5.2).

По мере ввода информации в строке запросов в поле чертежа будет появляться графическое изображение электрической схемы, которая в конечном виде будет выглядеть так, как показано на рис. 5.3.

Далее создадим функцию, которая вводит исходную информацию через аргументы (параметры функции). Для этого сформируем список данных по всем элементам схемы, включая проводники. Допустим, наша схема состоит из 12 элементов. Данные по каждому элементу схемы представляются в виде списка в списке LE. Первый элемент списка данных включает имя элемента, второй – номер элемента, третий – значение сопротивления,

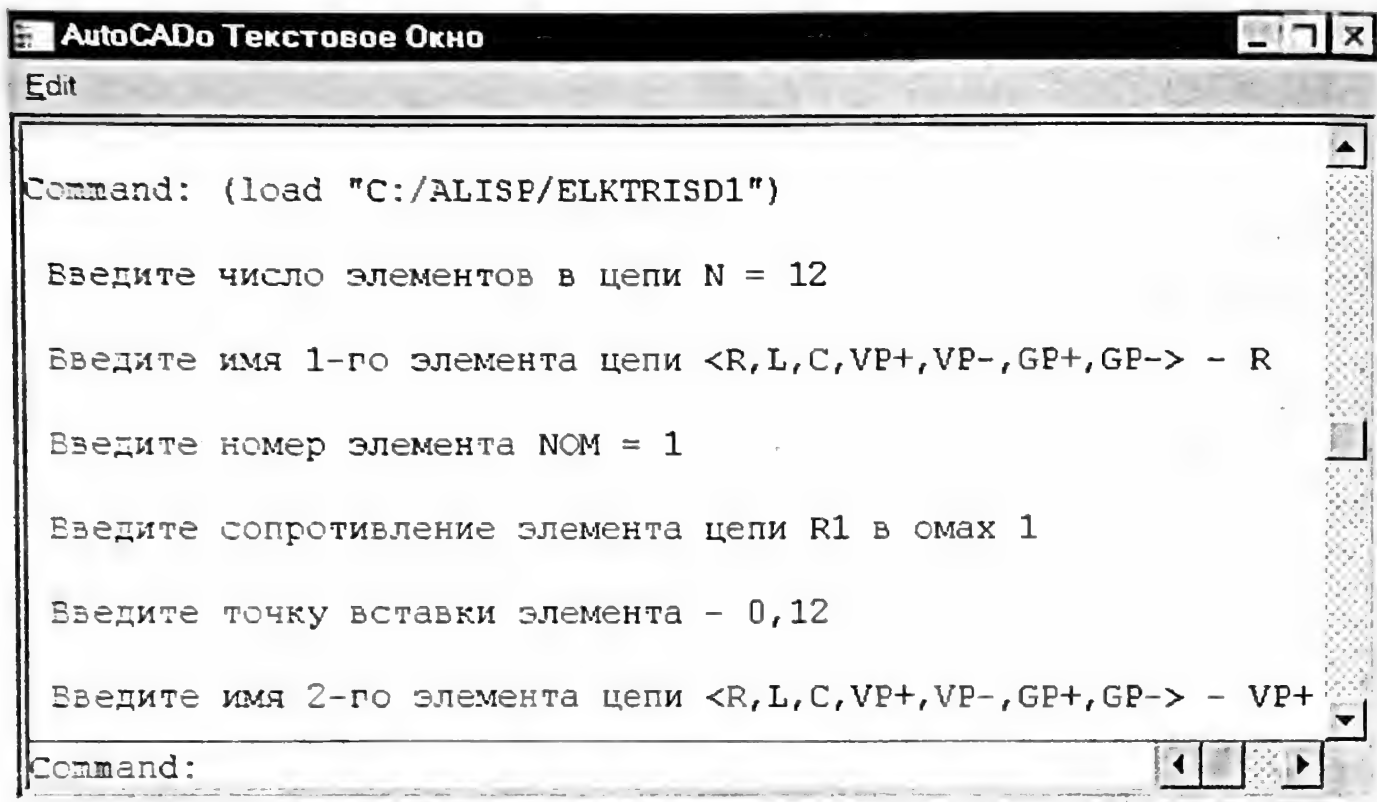


Рис. 5.2. Фрагмент данных для создания изображения электрической цепи в диалоговом режиме

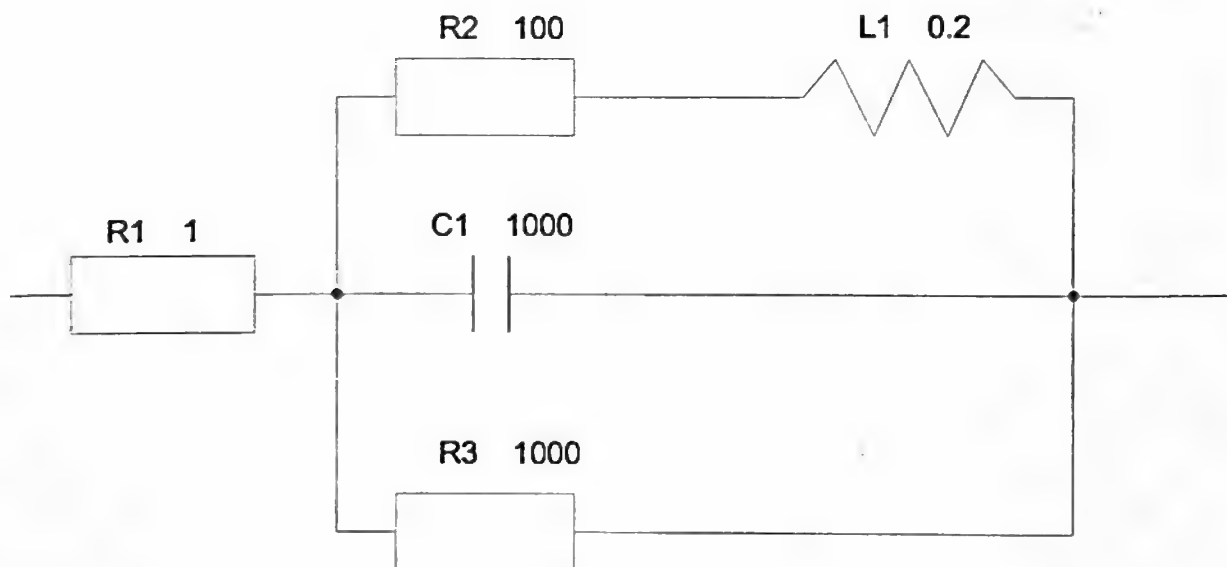


Рис. 5.3. Параметрическое изображение электрической схемы

четвертый – точку начала создания графического изображения элемента,
 пятый – точку окончания создания графического изображения элемента.

; Создание списка элементов цепи
 ; для графического изображения схемы

```
(SETQ LE (LIST=
  '("R" 1 1 (0 10) (10 10))
  '("VP+" 1 0 (10 10) (10 20))
  '("VP-" 1 0 (10 10) (10 0))
  '("R" 2 100 (10 20) (20 20))
  '("L" 1 0.2 (20 20) (30 20))
  '("C" 1 1000 (10 10) (20 10))
  '("R" 3 1000 (10 0) (20 0))
  '("GP+" 1 0 (20 10) (30 10))
  '("VP-" 2 1 (30 20) (30 10))
  '("VP+" 2 0 (30 0) (30 10))
  '("GP+" 2 0 (20 0) (30 0))
  '("GP+" 3 0 (30 10) (40 10)))
```

Для формирования графического изображения электрической схемы переменного тока, состоящей из сопротивлений, конденсаторов, катушек индуктивностей и проводников, создадим функцию (SHEMARIS...):

```
(DEFUN SHEMARIS (LL)
  (FOREACH EL LL
    (SETQ TB (NTH 3 EL)      ; точка вставки элемента
          X (NTH 0 TB)
          Y (NTH 1 TB)
          IMY (NTH 0 EL)    ; имя элемента
          NOM (NTH 1 EL)    ; номер элемента
          NUM (NTH 2 EL)    ; сопротивление элемента
    )
    (COND
      ((EQ IMY "R") (RP))    ; изображение резистора
      ((EQ IMY "L") (LP))    ; изображение катушки индуктивности
      ((EQ IMY "C") (CP))    ; изображение конденсатора
      ((EQ IMY "VP-") (VP-)) ; изобр. вертикального провода вниз
      ((EQ IMY "VP+") (VP+)) ; изобр. вертикального провода вверх
      ((EQ IMY "GP+") (GP+)) ; изобр. горизонтального провода вправо
      ((EQ IMY "GP-") (GP-)) ; изобр. горизонтального провода влево
      (T (RP)))
  )
```

В окончательном виде комплекс функций для создания параметрического изображения электрической схемы, состоящей из сопротивлений, конденсаторов и катушек индуктивностей, представлен ниже.

5.1.5. Программа параметрического изображения электрической схемы

```
; *****
; Комплекс функций параметрического изображения электрической
; схемы, состоящей из сопротивлений, конденсаторов и катушек
; индуктивностей
; *****

(SETVAR "BLIPMODE" 0)      ; отключение маркеров
(SETVAR "CMDECHO" 0)      ; отключение эха команд
(DEFUN RP ()              ; функция изображения резистора
  (COMMAND "PLINE" TB "W" "0.01" "0.01" "@2,0" "@0,1" "@6,0"
    "@0,-1" "@2,0" "@-2,0" "@0,-1" "@-6,0" "@0,1" "")
    "TEXT" (LIST (+ X 2) (+ Y 1.5)) "0.7" "0" "R"
    "TEXT" (LIST (+ X 3) (+ Y 1.5)) "0.7" "0" "NOM"
    "TEXT" (LIST (+ X 5) (+ Y 1.5)) "0.7" "0" "NUM")
  )
(DEFUN LP ()              ; функция изображения катушки индуктивности
  (COMMAND "PLINE" TB "W" "0.01" "0.01" "@2,0" "@1,1" "@1,-2"
    "@1,2" "@1,-2" "@1,2" "@1,-1" "@2,0" "")
    "TEXT" (LIST (+ X 2) (+ Y 1.5)) "0.7" "0" "L"
    "TEXT" (LIST (+ X 3) (+ Y 1.5)) "0.7" "0" "NOM"
    "TEXT" (LIST (+ X 5) (+ Y 1.5)) "0.7" "0" "NUM")
  )
(DEFUN CP ()              ; функция изображения конденсатора
  (COMMAND
    "PLINE" TB "W" "0.01" "0.01" "@4,0" "@0,1" "@0,-2" ""
    "PLINE" "@2,0" "W" "0.01" "0.01" "@0,2" "@0,-1" "@4,0" ""
    "TEXT" (LIST (+ X 2) (+ Y 1.5)) "0.7" "0" "C"
    "TEXT" (LIST (+ X 3) (+ Y 1.5)) "0.7" "0" "NOM"
    "TEXT" (LIST (+ X 5) (+ Y 1.5)) "0.7" "0" "NUM")
  )
(DEFUN VP+ ()             ; функция изображения вертикального
                          ; проводника вверх
  (COMMAND "PLINE" TB "W" "0.01" "0.01" "@0,10" "")
  )
(DEFUN VP- ()             ; функция изображения вертикального
                          ; проводника вниз
  (COMMAND "PLINE" TB "W" "0.01" "0.01" "@0,-10" "")
  )
; функция изображения горизонтального проводника вправо
(DEFUN GP+ () (COMMAND "PLINE" TB "W" "0.01" "0.01" "@10,0" ""))
)
```

```

; функция изображения горизонтального проводника влево
(DEFUN GP- ()
  (COMMAND "PLINE" TB "W" "0.01" "0.01" "@-10,0" "")
)
(COMMAND "LIMITS" "0,0" "40, 30")
(COMMAND "ZOOM" "A")
(DEFUN SHEMARIS (LL)
  (FOREACH EL LL
    (SETQ TB (NTH 3 EL) ; точка вставки элемента
      X (NTH 0 TB)
      Y (NTH 1 TB)
      IMY (NTH 0 EL) ; имя элемента
      NOM (NTH 1 EL) ; номер элемента
      NUM (NTH 2 EL) ; сопротивление элемента
    (COND ((EQ IMY "R") (RP)) ; изображение резистора
      ((EQ IMY "L") (LP)) ; изображение катушки индуктивности
      ((EQ IMY "C") (CP)) ; изображение конденсатора
      ((EQ IMY "VP-") (VP-)) ; изобр. вертикального провода вниз
      ((EQ IMY "VP+") (VP+)) ; изобр. вертикального провода вверх
      ((EQ IMY "GP+") (GP+)) ; изобр. горизонтального провода вправо
      ((EQ IMY "GP-") (GP-)) ; изобр. горизонтального провода влево
      (T (RP))))
  )

```

Этот комплекс функций целесообразно сохранить в файле, например, ELKTRISL.LSP. Допустим, этот файл находится в папке (каталоге) ALISP на диске С:.

Для создания в пакетном режиме графического изображения электрической схемы переменного тока, состоящей из сопротивлений, конденсаторов и катушек индуктивностей, необходимо загрузить указанный файл в командной строке AutoCAD:

Command: (LOAD "C:/ALISP/ELKTRISL")

Затем предстоит сформировать список LE. Например, для электрической схемы (см. рис. 5.3) список LE можно записать так:

```

(SETQ LE (LIST ; список элементов цепи для изображения схемы
  ("R" 1 1 (2 12) (12 12))
  ("VP+" 1 0 (12 12) (12 22))
  ("VP-" 1 0 (12 12) (12 2))
  ("R" 2 100 (12 22) (22 22))
  ("L" 1 0.2 (22 22) (32 22))
  ("C" 1 1000 (12 12) (22 12))
  ("R" 3 1000 (12 2) (22 2))

```



```

' ("GP+" 1 0 (22 12) (32 12))
' ("VP-" 2 1 (32 22) (32 12))
' ("VP+" 2 0 (32 2) (32 12))
' ("GP+" 2 0 (22 2) (32 2))
' ("GP+" 3 0 (32 12) (42 12))
)
)

```

и вызвать функцию:

```
(SHEMARIS LE)
```

Список LE можно ввести непосредственно в функцию (SHEMARIS...):

```

(SHEMARIS (LIST
' ("R" 1 1 (2 12) (12 12))
' ("VP+" 1 0 (12 12) (12 22))
' ("VP-" 1 0 (12 12) (12 2))
' ("R" 2 100 (12 22) (22 22))
' ("L" 1 0.2 (22 22) (32 22))
' ("C" 1 1000 (12 12) (22 12))
' ("R" 3 1000 (12 2) (22 2))
' ("GP+" 1 0 (22 12) (32 12))
' ("VP-" 2 1 (32 22) (32 12))
' ("VP+" 2 0 (32 2) (32 12))
' ("GP+" 2 0 (22 2) (32 2))
' ("GP+" 3 0 (32 12) (42 12))
)
)

```

В результате получим параметрическое изображение электрической схемы, представленной на рис. 5.3.

5.1.6. Основные этапы автоматизации расчета электрических схем переменного тока

Рассмотрим теперь основные этапы автоматизации расчета электрических схем переменного тока.

Постановка задачи

Требуется разработать систему для автоматизации расчета электрических схем переменного тока, состоящих из сопротивлений, индуктивностей и конденсаторов. Система должна быть объектно-ориентированной, предназначенной для описания и расчета электрических схем любой сложности. Расчетные параметры – полное сопротивление электрической схемы и резонансная частота.

Выявление основных особенностей, взаимосвязей и количественных закономерностей

Все элементы электрической схемы, как базовые, так и комбинации базовых элементов, могут быть соединены последовательно или параллельно, а также в разных сочетаниях.

Каждый базовый элемент электрической схемы можно представить в виде функции, для наименования которой целесообразно использовать обозначения, применяемые наиболее часто. Так, для расчета сопротивления резистора R создадим функцию ($R\ OM$), для расчета сопротивления катушки индуктивности L – ($L\ HEN$), а для конденсатора C – ($C\ FAR$). Аргументами функций являются: OM – сопротивление, измеренное в омах; HEN – индуктивность, измеренная в генри; FAR – емкость конденсатора, измеренная в фарадах.

Чтобы унифицировать запись полного сопротивления для базовых элементов, используем комплексное число, которое можно представить в виде списка, состоящего из вещественной части и мнимой части комплексного числа.

Полное сопротивление для резистора равно вещественной части, мнимая часть равна 0.

Полное сопротивление для катушки определяется только мнимой частью и равно $W * HEN$.

Полное сопротивление для конденсатора также определяется только мнимой частью и соответственно равно $-1 / (W * FAR)$.

Последовательное соединение двух элементов E_1 и E_2 будем описывать функцией ($POSL\ E1\ E2$).

Полное сопротивление последовательного соединения двух элементов определяется известной формулой сложения двух комплексных чисел:

$$Z = Z_1 + Z_2 = (A_1 + B_1 I) + (A_2 + B_2 I) = (A_1 + A_2) + (B_1 + B_2) I$$

Параллельное соединение двух элементов E_1 и E_2 будем описывать функцией ($PAR\ E1\ E2$).

Полное сопротивление параллельного соединения двух элементов определяется следующей формулой:

$$Z = 1 / (1 / Z_1 + 1 / Z_2)$$

Величину $1 / Z_1 = 1 / (A_1 + B_1 I)$ можно представить как

$$1 / (A_1 + B_1 I) = (A_1 - B_1 I) / (A_1^2 + B_1^2) = A_1 / (A_1^2 + B_1^2) - B_1 I / (A_1^2 + B_1^2)$$

Аналогичное выражение можно записать и для $1 / Z_2$.

Для поиска резонансной частоты необходимо, изменяя частоту переменного тока, найти такую частоту, при которой полное сопротивление электрической схемы будет равно нулю.

Разработка последовательности действий и комплекса функций для расчета элементов электрической схемы

Создадим функции, определяющие полное сопротивление базовых элементов схемы:

```
(DEFUN R (OM) ; определяет полное сопротивление резистора
  (LIST OM 0)
)
(DEFUN L (HEN) ; определяет полное сопротивление индуктивности
  (LIST 0 (* W HEN))
)
(DEFUN C (FAR) ; определяет полное сопротивление конденсатора
  (LIST 0 (/ -1 (* W FAR)))
)
```

Далее создадим функцию (**POSL...**), которая определяет полное сопротивление схемы, состоящей из двух последовательно соединенных элементов:

```
(DEFUN POSL (Z1 Z2) ; определяет сопротивление цепи из двух
  ; последовательно соединенных элементов
  (SETQ A (+ (NTH 0 Z1) (NTH 0 Z2))
        B (+ (NTH 1 Z1) (NTH 1 Z2)))
  (LIST A B))
```

Создадим теперь функцию (**POSLN...**), которая определяет полное сопротивление схемы, состоящей из нескольких последовательно соединенных элементов, представленных в виде списка.

```
(DEFUN POSLN (LN) ; определяет сопротивление цепи из
  ; N последовательно соединенных элементов
  (COND (( NULL LN) NIL)
        ((= (LENGTH LN) 1) (NTH 0 LN))
        ((= (LENGTH LN) 2) (POSL (EVAL (NTH 0 LN))
                                     (EVAL (NTH 1 LN))))
        (T (POSL (EVAL (NTH 0 LN)) (POSLN (CDR LN)))))
)
```

Перед созданием функции, определяющей полное сопротивление схемы, состоящей из двух параллельно соединенных элементов, создадим функцию для вычисления обратного значения комплексного числа. При этом используем ранее полученное выражение:

$$1 / Z = 1 / (A + BI) = A / (A^2 + B^2) - BI / (A^2 + B^2).$$

```
(DEFUN KO (Z) ; определяет обратное значение
  (SETQ A (NTH 0 Z) ; комплексного числа 1 / Z
```

```

B (NTH 1 Z)
SUM (+ (* A A) (* B B))
(LIST (/ A SUM) (* B (/ -1 SUM)))

```

Для определения полного сопротивления схемы, состоящей из двух параллельно соединенных элементов, применим известную формулу:

$$Z = 1 / (1 / Z_1 + 1 / Z_2)$$

Соответствующая функция будет выглядеть следующим образом:

```

(DEFUN PAR (Z1 Z2) ; определяет сопротивление цепи из двух
                  ; параллельно соединенных элементов
  (KO (POSL (KO Z1) (KO Z2)))
)

```

Далее создадим функцию (**PARN...**), которая определяет полное сопротивление схемы, состоящей из нескольких параллельно соединенных элементов, представленных в виде списка:

```

(DEFUN PARN (LN) ; определяет сопротивление цепи из
                 ; N параллельно соединенных элементов
  (COND ((NULL LN) NIL)
        ((= (LENGTH LN) 1) (NTH 0 LN))
        ((= (LENGTH LN) 2) (PAR (EVAL (NTH 0 LN))
                                   (EVAL (NTH 1 LN))))
        (T (PAR (EVAL (NTH 0 LN)) (PARN (CDR LN)))))
)

```

Теперь создадим функцию (**REZONANS...**) для расчета резонансной частоты, которая по начальной частоте W и приращению частоты DW определяет резонансную частоту системы:

```

(DEFUN REZONANS (PRIZN W DW) ; определяет резонансную частоту
  (SETQ S (EVAL CEP))
  (COND ((< (NTH 1 S) 0) (SETQ ZN '<))
        ((> (NTH 1 S) 0) (SETQ ZN '>))
        ((= (NTH 1 S) 0) (SETQ ZN '=')))
  )
  (WHILE ((EVAL ZN) (NTH 1 S) 0) ; цикл поиска оптимальной частоты
    (SETQ W (+ W DW)) ; изменяет частоту в цепи
    (SETQ S (EVAL CEP))) ; расчет электрической цепи
  (PRINC "\n\n Результаты расчета ")
  (PRINC "\n Собственная частота схемы, 1/C WO = " (PRINC W))
  (IF (= PRIZN 1) (PROGN (TEXTSCR)
    (PRINC "\n Полное сопротивление схемы S = " (PRINC S)))
    (PRINC))
)

```


Комплекс функций для расчета элементов электрической схемы

```
; *****
; Комплекс функций объектно-ориентированной системы для расчета
; резонансной частоты электрических схем переменного тока,
; включающих резисторы, конденсаторы и катушки индуктивности.
; Результаты расчета: полное сопротивление и резонансная частота для
; электрических схем переменного тока
; *****
; Представление электрической цепи в виде вложенных функций,
; например, (SETQ CEP '(POSL (R 1) (PARN '((POSL (R 100) (L 0.2))
; (C 1.0E-6) (R 10E6))))))
; (POSL E1 E2) - функция расчета сопротивления двух последовательно
; соединенных элементов E1 и E2
; (PAR E1 E2) - функция расчета сопротивления двух параллельно
; соединенных элементов E1 и E2
; (KO Z) - определение обратного значения комплексного числа
; (POSLN LN) - функция расчета сопротивления последовательно
; соединенных элементов, описываемых в списке LN
; (PARN LN) - функция расчета сопротивления параллельно
; соединенных элементов, описываемых в списке LN
; Список может выглядеть так: '((R 100) (L 0.2) (C 1.0E-6))
; (R OM) - функция определения сопротивления резистора
; (L HEN) - функция определения сопротивления катушки
; (C FAR) - функция определения сопротивления конденсатора,
; где: OM - полное сопротивление, ом
; HEN - полное сопротивление индуктивности, генри
; FAR - полное сопротивление емкости, фарады
; (REZONANS PRIZN W DW) - функция для расчета резонансной частоты
; PRIZN - признак: 0 - расчет механической системы, 1 - электрической
; W - начальная частота расчета, 1/C
; DW - приращение частоты, 1/C
; Результаты расчета:
; S - полное сопротивление электрической схемы в виде списка,
; включающего действительную и мнимую части, например,
; S = (100.39 19.0338)
; WO - собственная частота схемы (системы)
; *****
(DEFUN R (OM) ; определяет сопротивление резистора
  (LIST OM 0)
)
(DEFUN L (HEN) ; определяет сопротивление индуктивности
  (LIST 0 (* W HEN))
)
```

```

)
(DEFUN C (FAR) ; определяет сопротивление конденсатора
  (LIST 0 (/ -1 (* W FAR)))
)
(DEFUN POSL (Z1 Z2) ; определяет сопротивление цепи из двух
  ; последовательно соединенных элементов
  (SETQ A (+ (NTH 0 Z1) (NTH 0 Z2))
    B (+ (NTH 1 Z1) (NTH 1 Z2)))
  (LIST A B)
)
(DEFUN POSLN (LN) ; определяет сопротивление цепи из N
  ; последовательно соединенных элементов
  (COND (( NULL LN) NIL)
    ((= (LENGTH LN) 1) (NTH 0 LN))
    ((= (LENGTH LN) 2) (POSL (EVAL (NTH 0 LN))
      (EVAL (NTH 1 LN)))))
    (T (POSL (EVAL (NTH 0 LN)) (POSLN (CDR LN)))))
)
(DEFUN KO (Z) ; определяет обратное значение
  ; комплексного числа Z (1/Z)
  (SETQ A (NTH 0 Z)
    B (NTH 1 Z)
    SUM (+ (* A A) (* B B)))
  (LIST (/ A SUM) (* B (/ -1 SUM)))
)
(DEFUN PAR (Z1 Z2) ; определяет сопротивление цепи из двух
  ; параллельно соединенных элементов
  (KO (POSL (KO Z1) (KO Z2)))
)
(DEFUN PARN (LN) ; определяет сопротивление цепи из
  ; N параллельно соединенных элементов
  (COND (( NULL LN) NIL)
    ((= (LENGTH LN) 1) (NTH 0 LN))
    ((= (LENGTH LN) 2) (PAR (EVAL (NTH 0 LN))
      (EVAL (NTH 1 LN)))))
    (T (PAR (EVAL (NTH 0 LN)) (PARN (CDR LN)))))
)
(DEFUN REZONANS (PRIZN W DW) ; определяет резонансную частоту
  (SETQ S (EVAL CEP))
  (COND ((< (NTH 1 S) 0) (SETQ ZN '<))
    ((> (NTH 1 S) 0) (SETQ ZN '>))
    ((= (NTH 1 S) 0) (SETQ ZN '=')))
  (WHILE ((EVAL ZN) (NTH 1 S) 0) ; цикл поиска оптимальной частоты

```

```

      (SETQ W (+ W DW))           ; изменение частоты в цепи
      (SETQ S (EVAL CEP)))        ; расчет электрической цепи
      (PRINC "\n\n Результаты расчета ")
      (PRINC "\n Собственная частота схемы, 1/C WO = ") (PRINC W)
      (IF (= PRIZN 1) (PROGN (TEXTSCR)
      (PRINC "\n Полное сопротивление схемы S = ") (PRINC S)))
      (PRINC)
    )
; Электрическую цепь представим на следующем примере:
(SETQ CEP '(POSL (R 1) (PARN '(((POSL (R 100) (L 0.2))
      (C 1.0E-6) (R 10E6)))))
(REZONANS 1 2000 1.0)           ; функция расчета резонансной частоты

```

Для расчета полного сопротивления электрической схемы и резонансной частоты необходимо создать файл, в котором находится вся исходная информация для расчета и описания электрической схемы. Назовем этот файл ELEKTRR.LSP. После загрузки файла в окно текстового редактора Visual LISP произведем запуск программы, щелкнув по кнопке **Load active edit window** (Загрузить текст активного окна редактирования). В окне консоли появятся результаты компиляции функций AutoLISP. Если программа набрана без ошибок, на экране появится сообщение о загрузке выражений (форм). После подсказки консоли наберем схему электрической цепи и вызовем для выполнения функцию (**REZONANS...**) (рис. 5.4).

После нажатия клавиши **Enter** (Ввод) начнется вычисление функций, и через некоторое время в окне консоли появятся результаты расчета (рис. 5.5).

Результаты расчета будут представлены также и в текстовом окне редактора AutoCAD.

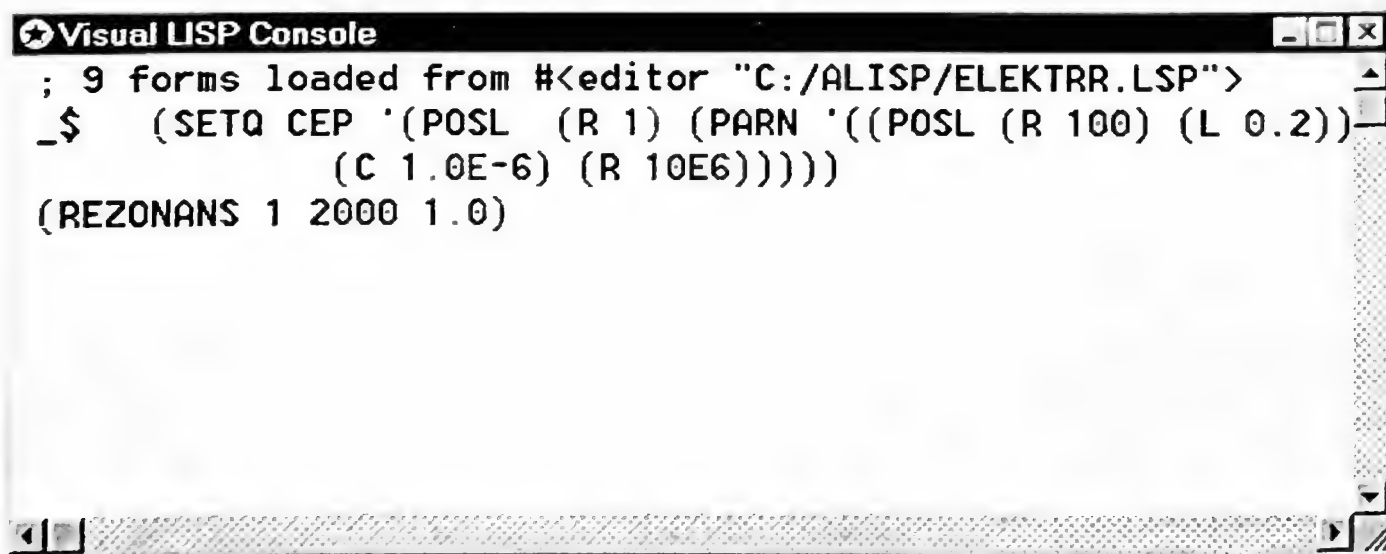
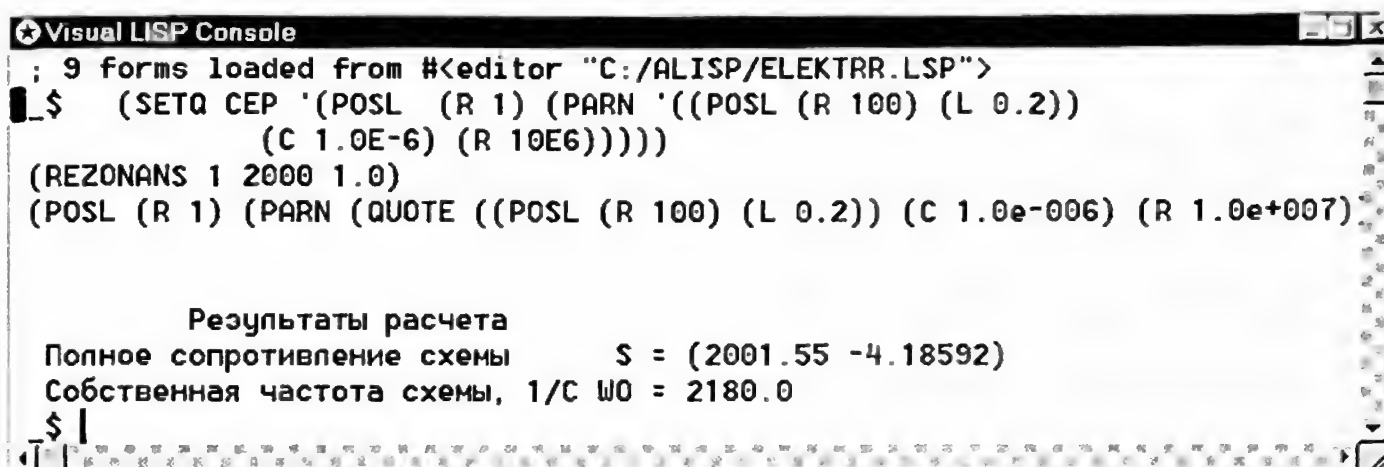


Рис. 5.4. Окно консоли с сообщением о результатах загрузки



```

Visual LISP Console
; 9 forms loaded from #<editor "C:/ALISP/ELEKTRR.LSP">
_$ (SETQ CEP '(POSL (R 1) (PARN '((POSL (R 100) (L 0.2))
      (C 1.0E-6) (R 10E6)))))
(REZONANS 1 2000 1.0)
(POSL (R 1) (PARN (QUOTE ((POSL (R 100) (L 0.2)) (C 1.0e-006) (R 1.0e+007)))))

Результаты расчета
Полное сопротивление схемы      S = (2001.55 -4.18592)
Собственная частота схемы, 1/C W0 = 2180.0
$ |

```

Рис. 5.5. Окно консоли с результатами расчета.

5.1.7. Основные этапы создания объектно-ориентированной подсистемы для расчета собственной частоты механической системы

Рассмотрим теперь основные этапы создания объектно-ориентированной подсистемы для расчета собственной частоты механической системы.

Ограничимся расчетом механических систем с конечным числом свободы, линейных, автономных, стационарных, консервативных и диссипативных, а также систем со свободными колебаниями.

Постановка задачи

Требуется создать объектно-ориентированную систему для описания и расчета механических систем любой сложности, состоящих из масс, упругих элементов и демпферов. Расчетным параметром должна быть собственная частота системы.

Выявление основных особенностей, взаимосвязей и количественных закономерностей

Все элементы механической системы, как базовые, так и комбинации базовых элементов, могут быть соединены последовательно, параллельно либо в разных комбинациях.

Для решения задачи используем аналогии, при которых физические процессы в оригинале и модели протекают по законам математического изоморфизма, то есть подчиняются одинаковым математическим законам, несмотря на то, что физическая природа процессов может быть различной. В качестве модели используем электрическую цепь переменного тока, которую мы уже умеем рассчитывать.

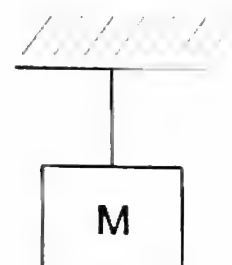


Рис. 5.6. Механическая система

Примером послужит одна из самых распространенных механических систем – кабина лифта, подвешенная на стальных тросах (рис. 5.6).

Кабина лифта массой $M = 450$ кг подвешена на канате. Длина каната $L = 16$ м. Площадь поперечного сечения каната $S = 16$ см². Модуль упругости материала каната $E = 2 \cdot 10^6$ кг/см².

Данная система может совершать колебания в вертикальной плоскости относительно центра статического равновесия, вес троса не учитываем.

Уравнение движения данной системы определяется уравнением д'Аламбера:

$$F_m + F_n = 0 \text{ или } M \frac{d^2 x}{dt^2} + Kx = 0$$

где F_m , F_n – сила инерции от массы и сила упругости каната соответственно;

x – перемещение от центра статического равновесия;

K – коэффициент жесткости упругого элемента.

Моделью такой системы может служить электрическая схема, которая представляет собой электрическую цепь, имеющую катушку индуктивности L и конденсатор C (рис. 5.7).

На основании II закона Кирхгофа э.д.с. должна быть равна сумме падений напряжений на элементах контура.

$$U_L + U_C = 0 \text{ или } L \frac{dI}{dt} + \frac{Q}{C} = 0$$

где U_L , U_C – падение напряжения на катушке индуктивности и конденсаторе соответственно;

I – переменный ток, протекающий в контуре.

t – время протекания процесса.

Следует учитывать, что скорость изменения заряда конденсатора представляет собой ток, то есть

$$I = \frac{dQ}{dt} \text{ или } \frac{dI}{dt} = \frac{d^2 Q}{dt^2}$$

Если сравнить математические модели механической системы и электрической цепи, можно увидеть, что уравнения математически изоморфны. Следовательно, электрическая

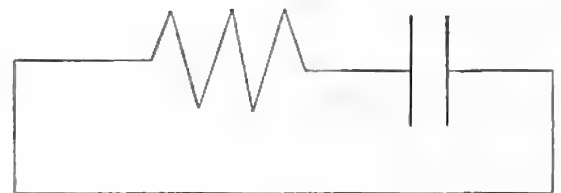


Рис. 5.7. Электрическая цепь, моделирующая лифт

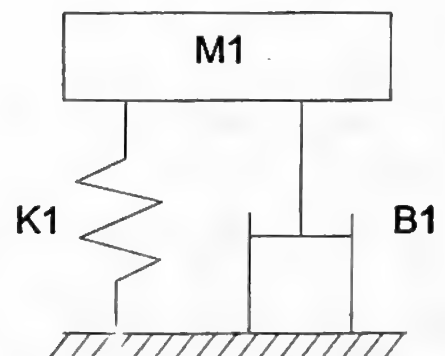


Рис. 5.8. Механическая система, моделирующая виброплощадку

цепь (рис. 5.7) является моделью-аналогом механической системы (рис. 5.6). При этом индуктивность L является аналогом массы M , емкость C – коэффициента упругости $1/K$, электрический ток I – аналогом скорости

$$V = dx/dt$$

В качестве другого примера рассмотрим вибрационную площадку (рис. 5.8). Сосредоточенная масса M_1 , включающая корпус виброплощадки и вибрируемые изделия, опирается на пружины с коэффициентом жесткости K_1 . В систему включены также жидкостные амортизаторы с коэффициентом вязкого трения B_1 .

Данная система может совершать колебания в вертикальной плоскости относительно центра статического равновесия,

Движение данной системы определяется уравнением д'Аламбера:

$$F_M + F_B + F_n = 0 \text{ или } M_1 \frac{d^2x}{dt^2} + B_1 \frac{dx}{dt} + K_1 x = 0$$

где F_M , F_B , F_n – сила инерции от массы, сила вязкого трения амортизаторов и сила упругости пружин соответственно;

x – перемещение от центра статического равновесия.

Моделью такой системы может служить электрическая схема, которая представляет собой электрический контур, имеющий сопротивление R_1 , катушку индуктивности L_1 и конденсатор C_1 с зарядом Q_1 (рис. 5.9).

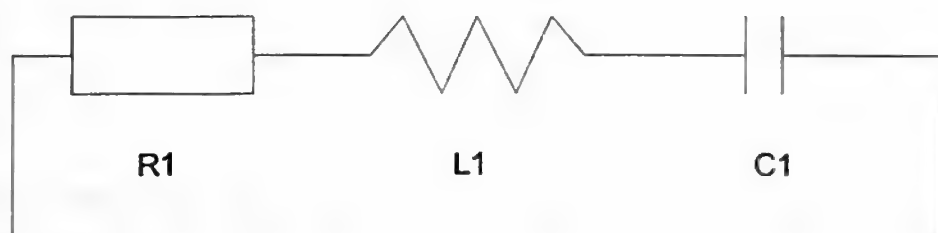


Рис. 5.9. Электрическая цепь, моделирующая виброплощадку

На основании II закона Кирхгофа э.д.с. должна быть равна сумме падений напряжений на элементах контура.

$$U_L + U_R + U_C = 0 \text{ или } L_1 \frac{dI}{dt} + R_1 I + \frac{Q_1}{C_1} = 0,$$

где U_L , U_R , U_C – падение напряжения на катушке индуктивности, сопротивлении и конденсаторе, соответственно;

I – переменный ток, протекающий в контуре;

t – время протекания процесса.

Необходимо учитывать, что скорость изменения заряда конденсатора представляет собой ток, т.е.

$$I = \frac{dQ}{dt}; \quad \frac{dI}{dt} = \frac{d^2Q_1}{dt^2} \quad \text{или} \quad L_1 \frac{d^2Q_1}{dt^2} + R_1 \frac{dQ_1}{dt} + \frac{Q_1}{C_1} = 0$$

Если сравнить математические модели механической системы и электрической цепи, можно увидеть, что уравнения математически изоморфны. Следовательно, электрическая цепь (рис. 5.9) является моделью-аналогом механической системы (рис. 5.8). При этом индуктивность L_1 – аналог массы M_1 , сопротивление R_1 – коэффициента вязкого трения B_1 , емкость C_1 – аналог коэффициента упругости $1/K_1$.

В силу изоморфности моделей механической системы и электрической цепи их можно исследовать на электрической модели, которую мы уже умеем рассчитывать.

Разработка последовательности действий и комплекса функций для расчета элементов электрической схемы

Для расчета колебательных механических систем с помощью электрических схем необходимо выполнить следующие основные действия:

- определить эквивалентную электрическую цепь;
- определить параметры эквивалентных электрических элементов;
- сделать расчет электрической схемы;
- произвести пересчет результатов моделирования.

На этапе определения эквивалентной электрической цепи моделируемая механическая система представляется в виде совокупности масс, упругих элементов и демпферов, связанных друг с другом, как в моделируемой системе (см. рис. 5.6).

Зная, что катушка индуктивности является аналогом массы, сопротивление – аналогом демпфера, а емкость – аналогом упругого элемента, можно составить эквивалентную электрическую цепь. Пример такой эквивалентной электрической цепи представлен на рис. 5.7.

Параметры основных элементов электрической цепи определяются следующим образом:

- индуктивность катушки L_1 в генри (Г) равна массе M в кг;
- емкость конденсатора C_1 в фарадах (Ф) равна обратной величине коэффициента жесткости $1/K$ в Н/м;
- сопротивление резистора R_1 в омах равно величине коэффициента вязкого трения B_1 в кг/с.

Теперь определим резонансную частоту механической системы, представленной на рис. 5.8. В ходе расчета необходимо определить эквивалентную

электрическую цепь (для механической системы, представленной на рис. 5.8, она уже определена и изображена на рис. 5.9), а также параметры эквивалентных электрических элементов. Параметры элементов электрической схемы (рис. 5.9) вычисляются с учетом заранее определенных переходных масштабов:

индуктивность $L1 = 450$ Гн;

коэффициент жесткости для каната определяется по формуле:

$$K_1 = E * S / L = 1 * 10^{10} * 0,0016 / 18 = 888889,0 \text{ Н/м}$$

где E – модуль упругости каната, кг/м²;

S – площадь сечения каната, м².

Тогда емкость C_1 будет равна:

$$C_1 = 1 / K_1 = 1 / 888889,0 = 1,125E-06 \text{ Ф.}$$

Для расчета данной электрической схемы воспользуемся программой ELEKTRR.LSP, разработанной в предыдущем параграфе.

Вначале создадим файл, в котором будет собрана вся исходная информация для расчета и представления электрической схемы-аналога. Назовем этот файл DIN1.LSP. Одновременно произведем загрузку объектно-ориентированной системы для расчета полного сопротивления и резонансной частоты электрической цепи (файл ELEKTRR.LSP).

При использовании системы Visual LISP файл ELEKTRR.LSP может находиться в одном из текстовых окон интегрированной среды Visual LISP. После успешного запуска функций из окна текстового редактора (кнопка **Load active edit window** (Загрузить текст активного окна редактирования)) на экране появится сообщение о загрузке программы в окне консоли. В этом же окне необходимо ввести исходную информацию о системе.

```
(SETQ M1 450.0           ; масса лифта в кг
      LK 18.0           ; длина каната в м
      SK 1.6E-03        ; площадь сечения каната в кв.м
      EK 1.0E+10        ; модуль упругости каната, кг/кв.м
      K1 (/ (* EK SK) LK) ; жесткость каната, Н/м
      L1 M1             ; индуктивность катушки, генри
      C1 (/ 1 K1)       ; емкость конденсатора, фарады
)
```

Механическая система – лифт в виде эквивалентной электрической цепи – может быть представлена так:

```
(SETQ CEP '(POSL (L L1) (C C1)))
```

Затем обратимся к функции расчета резонансной частоты систем:

```
(REZONANS 0 1 0.1) ; функция расчета резонансной частоты
```

Результаты расчета резонансной частоты лифта представлены на рис. 5.10.


```

Visual LISP Console
_$
; 9 forms loaded from #<editor "C:/ALISP/ELEKTRR.LSP">
_$ (SETQ PRIZN 0 ; 0 - расчет механической системы
    M1 450.0 ; масса лифта в кг
    LK 18.0 ; длина каната в м
    SK 1.6E-03 ; площадь сечения каната в кв. м
    EK 1.0E+10 ; модуль упругости каната кг/кв. м
    K1 (/ (* EK SK) LK) ; жесткость каната Н/м
    L1 M1 ; индуктивность катушки, генри
    C1 (/ 1 K1) ; емкость конденсатора, фарады
)
(SETQ CEP *(POSL (L L1) (C C1)))
(REZONANS 0 1 0.1)
1.125e-006
(POSL (L L1) (C C1))

Собственная частота схемы, 1/C W0 = 44.5
_$

```

Рис. 5.10. Результаты расчета резонансной частоты лифта

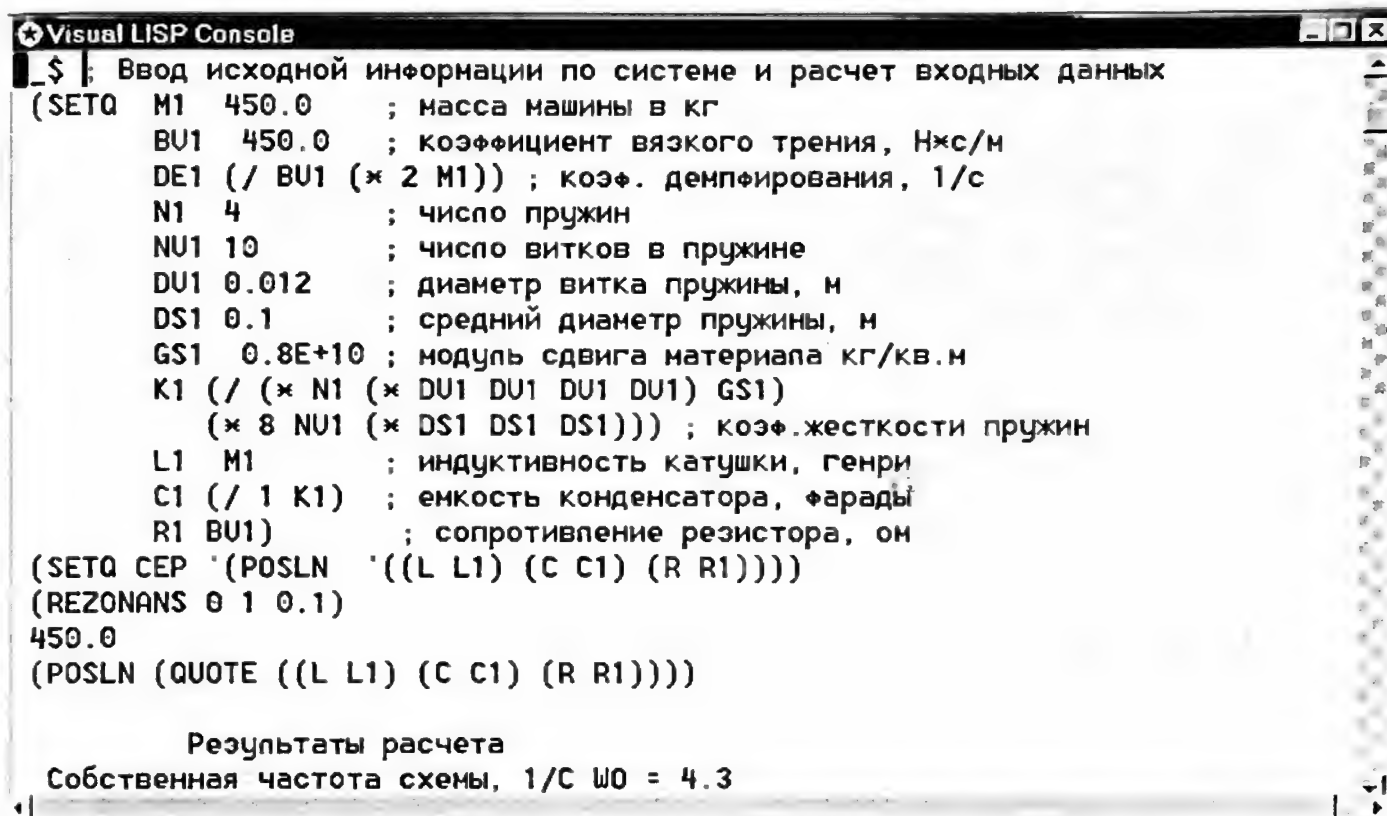
Для расчета резонансной частоты виброплощадки необходимо создать файл, в котором будет находиться вся исходная информация для расчета и описания самой электрической схемы-аналога. Назовем этот файл DIN2.LSP. Одновременно следует произвести загрузку объектно-ориентированной системы для расчета полного сопротивления и резонансной частоты электрической цепи (файл ELEKTRR.LSP).

При использовании системы Visual LISP файл ELEKTRR.LSP может находиться в одном из текстовых окон интегрированной среды Visual LISP. После успешного запуска функций из окна текстового редактора (кнопка **Load active edit window** (Загрузить текст активного окна редактирования)) на экране появится сообщение о загрузке программы в окне консоли. В этом же окне необходимо ввести исходную информацию о системе, например, следующую:

```

(SETQ
  M1 450.0 ; масса машины в кг
  BV1 450.0 ; коэффициент вязкого трения, Н*с/м
  DE1 (/ BV1 (* 2 M1)) ; коэффициент демпфирования, 1/с
  N1 4 ; число пружин
  NV1 10 ; число витков в пружине
  DV1 0.012 ; диаметр витка пружины, м
  DS1 0.1 ; средний диаметр пружины, м
  GS1 0.2E+10 ; модуль сдвига материала, кг/кв.м
  K1 (/ (* N1 (* DV1 DV1 DV1 DV1) GS1)

```



```

Visual LISP Console
_$ : Ввод исходной информации по системе и расчет входных данных
(SETQ M1 450.0 ; масса машины в кг
      BV1 450.0 ; коэффициент вязкого трения, Н*с/м
      DE1 (/ BV1 (* 2 M1)) ; коэф. демпфирования, 1/с
      N1 4 ; число пружин
      NU1 10 ; число витков в пружине
      DU1 0.012 ; диаметр витка пружины, м
      DS1 0.1 ; средний диаметр пружины, м
      GS1 0.8E+10 ; модуль сдвига материала кг/кв.м
      K1 (/ (* N1 (* DU1 DU1 DU1 DU1) GS1)
            (* 8 NU1 (* DS1 DS1 DS1))) ; коэф. жесткости пружин
      L1 M1 ; индуктивность катушки, Генри
      C1 (/ 1 K1) ; емкость конденсатора, фарады
      R1 BV1) ; сопротивление резистора, ом
(SETQ CEP '(POSLN '((L L1) (C C1) (R R1))))
(REZONANS 0 1 0.1)
450.0
(POSLN (QUOTE ((L L1) (C C1) (R R1))))

Результаты расчета
Собственная частота схемы, 1/С W0 = 4.3

```

Рис. 5.11. Результаты расчета резонансной частоты виброплощадки

```

(* 8 NV1 (* DS1 DS1 DS1))) ; коэффициент жесткости пружин
L1 M1 ; индуктивность катушки, генри
C1 (/ 1 K1) ; емкость конденсатора, фарады
R1 BV1) ; сопротивление резистора, ом
(SETQ CEP '(POSLN '((L L1) (C C1) (R R1)))) ; описание системы
(REZONANS 0 1 0.1) ; вызов функции расчета резонансной частоты

```

Расчет резонансной частоты виброплощадки представлен на рис. 5.11.

5.2. Сетевое планирование и управление проектами

Рассмотрим основные этапы создания системы сетевого планирования и управления проектами, включая создание параметрического изображения сетевого графика и результатов расчета в различных режимах. Начнем с создания графического изображения сетевого графика.

5.2.1. Постановка задачи

Разработать комплекс функций для автоматизации создания параметрического изображения сетевого графика с заданным числом событий и работ. На сетевом графике должна быть представлена вся исходная информация для расчета сетевого графика. Один из таких сетевых графиков изображен на рис. 5.12.

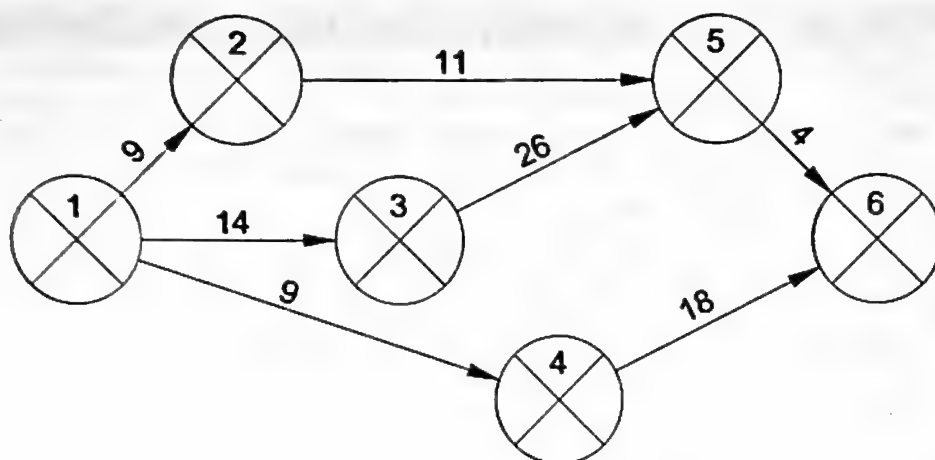


Рис. 5.12. Сетевой график выполнения проекта

В комплексе функций должна быть предусмотрена возможность изменения размера поля для создания графического изображения сетевого графика, размера кружочка, изображающего событие.

Необходимая исходная информация запрашивается в диалоговом режиме и вводится с клавиатуры компьютера.

5.2.2. Разработка последовательности действий и комплекса функций параметрического изображения элементов сетевого графика

Для создания параметрического изображения сетевого графика предварительно необходимо определить размер рисунка, число изображаемых на нем событий (кружочков), а также размер кружочка. Вся эта информация должна быть введена в диалоговом режиме. Высота текста может быть определена, например, как 0.3 от радиуса кружочка. Если информация не вводится, используется значение по умолчанию. Обычно информация по умолчанию указывается в угловых скобках, например, <20>

```

(SETQ BH (GETPOINT "\n Введите размер рисунка, <280,210>: "))
NS (GETINT "\n Введите число событий, NS = ")
R (GETREAL "\n Введите радиус кружочка, <20>: "))

(SETVAR "CMDECHO" 0)
(IF (EQ BH NIL) (SETQ BH (LIST 280 210)))
(IF (EQ R NIL) (SETQ R 20)) ; радиус кружочка по умолчанию
(SETQ H (* R 0.3)) ; высота цифр на сетевом графике
  
```

Если проанализировать рис. 5.12, на котором представлен сетевой график, можно заметить, что кружочки разделены на сектора и в верхнем указан номер события.

Создание изображения кружочков в сетевом графике целесообразно оформить в виде цикла:

```
(SETQ K 0 ; определение номера события
  LS '()) ; список координат событий
(REPEAT NS ; цикл ввода координат событий
  (SETQ K (+ 1 K)) ; определение номера вводимого события
  (SETVAR "CMDECHO" 1)
  (SETQ TR (GETPOINT (STRCAT "\n Введите координаты "
    (ITOA K) "-го события в виде X,Y [можно мышкой] ")))
  (SETQ LS (CONS (LIST K TR) LS))
  (SOBN K)
)
```

При этом графическое изображение кружочка с секторами можно оформить в виде функции (**SOBN...**):

```
(DEFUN SOBN (N) ; функция изображения кружочка
  (SETVAR "CMDECHO" 0) ; с разделением его на сектора
  (SETQ TN (ITOA N)
    TT (LIST (- (CAR TR) H) (+ (CAR (CDR TR)) H)))
  (COMMAND "CIRCLE" TR R
    "LINE" TR (POLAR TR (* PI 0.25) R) ""
    "LINE" TR (POLAR TR (* PI 0.75) R) ""
    "LINE" TR (POLAR TR (* PI 1.25) R) ""
    "LINE" TR (POLAR TR (* PI 1.75) R) ""
    "TEXT" TT H "0" TN ) ; ввод номера события
)
```

Исходя из логики выполнения проекта и наглядности его представления, пользователь определяет местоположения кружочков (событий) путем ввода соответствующих координат X и Y или с помощью мышки. Например, для события под номером 1 могут быть введены координаты: 30,100.

Следующим шагом будет ввод на сетевом графике числа операций (работ) и их изображение с указанием продолжительности выполнения каждой работы. При этом стрелки должны быть взаимосвязаны с расположением кружочков (событий). Следовательно, для каждой работы (стрелки) определяют координаты события, из которого выходит работа, и координаты события, в которое она входит, а также угол наклона стрелки и место изображения продолжительности выполнения работы.

Ввод данных по работам целесообразно оформить в виде цикла. При этом указывается событие I, из которого выходит работа, и событие J, в которое она входит, а затем указывается продолжительность этой работы TIJ:

```
(SETQ NR (GETINT "\n Введите число работ в сети NR = ")
  M 0)
(REPEAT NR
  (SETVAR "CMDECHO" 1)
```

```

(SETQ M (+ M 1) ; определяет номер вводимой работы
  TIJ (GETPOINT (STRCAT "\n Введите продолжительность "
    (ITOA M) "-ой работы в виде I,J,T : ")))
(SETVAR "CMDECHO" 0)
(SETQ II (FIX (NTH 0 TIJ))
  JJ (FIX (NTH 1 TIJ))
  TRI (NTH 1 (ASSOC II LS))
  TRJ (NTH 1 (ASSOC JJ LS))
  U (ANGLE TRI TRJ)
  UGR (* U (/ 180 PI))
  D (DISTANCE TRI TRJ)
  TTT (POLAR TRI U (* D 0.4))
  TRI (POLAR TRI U R)
  TRJ (POLAR TRJ U (- 0 R))
  TRJ1 (POLAR TRJ U (- 0 (* 0.5 R)))
  TX (RTOS (NTH 2 TIJ) 2 0))
(COMMAND "LINE" TRI TRJ1 ""
  "PLINE" TRJ1 "W" (/ R 5) (/ R 25) TRJ ""
  "TEXT" TTT H UGR TX)
)

```

В окончательном виде программа для создания в диалоговом режиме графического изображения сетевого графика представлена ниже.

5.2.3. Комплекс функций параметрического изображения сетевого графика в диалоговом режиме

```

; *****
; Комплекс функций для создания в диалоговом режиме
; параметрического изображения сетевого графика
; *****
(SETVAR "CMDECHO" 1)
(SETQ BH (GETPOINT "\n Введите размер рисунка, <280,210>: ")
  NS (GETINT "\n Введите число событий, NS = ")
  R (GETREAL "\n Введите радиус кружочка, <20>: "))
(SETVAR "CMDECHO" 0)
(IF (EQ BH NIL) (SETQ BH (LIST 280 210)))
(IF (EQ R NIL) (SETQ R 20)) ; радиус кружочка по умолчанию
(SETQ H (* R 0.3)) ; высота цифр на сетевом графике
(DEFUN SOBN (N) ; функция изображения кружочка
  (SETVAR "CMDECHO" 0) ; с разделением на секторы
  (SETQ TN (ITOA N)
    TT (LIST (- (CAR TR) H) (+ (CAR (CDR TR)) H)))
  (COMMAND "CIRCLE" TR R

```



```

"LINE" TR (POLAR TR (* PI 0.25) R) ""
"LINE" TR (POLAR TR (* PI 0.75) R) ""
"LINE" TR (POLAR TR (* PI 1.25) R) ""
"LINE" TR (POLAR TR (* PI 1.75) R) ""
"TEXT" TT H "0" TN ) ; ввод номера события
)
(COMMAND "LIMITS" "0,0" BH "ZOOM" "A")
(SETVAR "BLIPMODE" 0) ; отключение маркеров
(SETQ K 0 ; определение номера события
  LS '()) ; список координат событий
(REPEAT NS ; цикл ввода координат событий
  (SETQ K (+ 1 K))
  (SETVAR "CMDECHO" 1)
  (SETQ TR (GETPOINT (STRCAT "\n Введите координаты "
    (ITOA K) "-го события в виде X,Y [можно мышкой] ")))
  )
  (SETQ LS (CONS (LIST K TR) LS))
  (SOBN K) ; вызов функции изображения кружочка (события)
)
(SETQ NR (GETINT "\n Введите число работ в сети NR = ")
  M 0)
(REPEAT NR
  (SETVAR "CMDECHO" 1)
  (SETQ M (+ M 1)
    TIJ (GETPOINT (STRCAT "\n Введите продолжительность "
      (ITOA M) "-й работы в виде I,J,T ")))
  (SETVAR "CMDECHO" 0)
  (SETQ II (FIX (NTH 0 TIJ))
    JJ (FIX (NTH 1 TIJ))
    TRI (NTH 1 (ASSOC II LS))
    TRJ (NTH 1 (ASSOC JJ LS))
    U (ANGLE TRI TRJ)
    UGR (* U (/ 180 PI))
    D (DISTANCE TRI TRJ)
    TTT (POLAR TRI U (* D 0.4))
    TRI (POLAR TRI U R)
    TRJ (POLAR TRJ U (- 0 R))
    TRJ1 (POLAR TRJ U (- 0 (* 0.5 R)))
    TX (RTOS (NTH 2 TIJ) 2 2))
  (COMMAND "LINE" TRI TRJ1 ""
    "PLINE" TRJ1 "W" (/ R 5) (/ R 25) TRJ ""
    "TEXT" TTT H UGR TX)
)

```

Для создания в диалоговом режиме графического изображения сетевого графика достаточно обратиться к комплексу функций, созданных выше. Данный комплекс функций целесообразно сохранить в файле, например, SETRIS.LSP. Допустим, этот файл находится в папке (каталоге) ALISP на диске C:.

После загрузки этого файла в командной строке AutoCAD

```
Command: (LOAD "C:/ALISP/SETRIS")
```

будет запрашиваться информация, необходимая для создания графического изображения сетевого графика. В качестве примера возьмем сетевой график, представленный на рис. 5.12. На запросы в командной строке AutoCAD введем данные, представленные ниже в правой части запросов. Все запросы и данные по ним будут отражаться в текстовом окне AutoCAD (Shift+F2).

Вводимые пользователем значения выделим жирным шрифтом. После введения значений по каждому запросу необходимо нажать клавишу Enter (Ввод).

Введите размер рисунка, <280,210>:

Введите число событий, NS = 6

Введите радиус кружочка, <20>:

Введите координаты 1-го события: 20,100

Введите координаты 2-го события: 70,150

Введите координаты 3-го события: 120,100

Введите координаты 4-го события: 170,50

Введите координаты 5-го события: 220,150

Введите координаты 6-го события: 270,100

Введите число работ в сети NR = 7

Введите продолжительность 1-й работы в виде I,J,T : 1,2,8.8

Введите продолжительность 2-й работы в виде I,J,T : 1,3,14

Введите продолжительность 3-й работы в виде I,J,T : 1,4,9

Введите продолжительность 4-й работы в виде I,J,T : 2,5,11.3

Введите продолжительность 5-й работы в виде I,J,T : 3,5,25.7

Введите продолжительность 6-й работы в виде I,J,T : 4,6,17.7

Введите продолжительность 7-й работы в виде I,J,T : 5,6,3.5

Фрагмент введенных в диалоговом режиме данных для создания графического изображения сетевого графика представлен в текстовом окне AutoCAD (Shift+F2) на рис. 5.13.

По мере ввода информации в строке запросов в поле чертежа будет появляться графическое изображение сетевого графика, который в окончательном виде будет выглядеть, как на рис. 5.12.

А сейчас несколько усложним задачу. Допустим, мы имеем программу и результаты расчета этого же сетевого графика, однако получены они с применением другого языка программирования, например, Fortran.

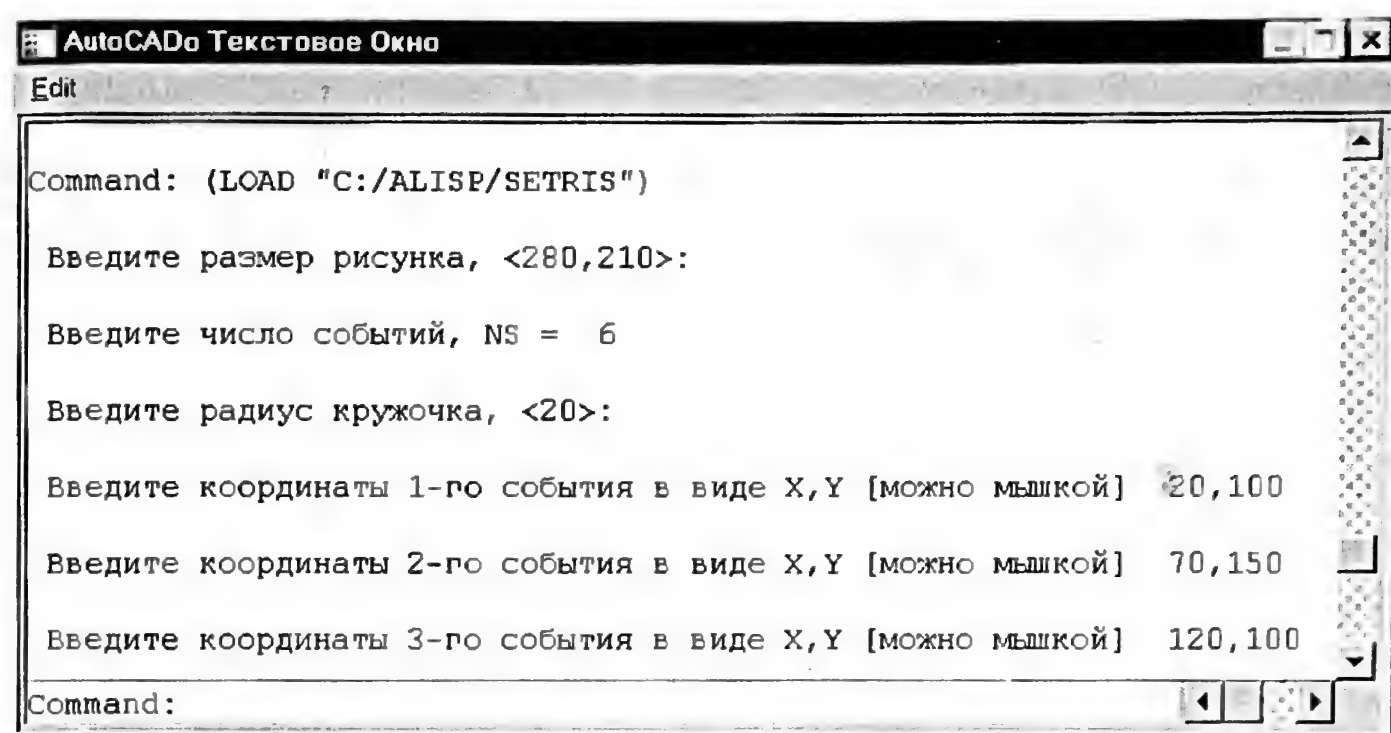


Рис. 5.13. Фрагмент введенных в текстовом окне AutoCAD данных для построения в диалоговом режиме изображения сетевого графика

5.2.4. Разработка комплекса функций создания изображения сетевого графика по результатам расчета его на языке Fortran

Предстоит разработать комплекс функций AutoLISP для создания графического изображения сетевого графика в среде AutoCAD по результатам расчета его на языке Fortran. Вся исходная информация вводится из выходного файла SETD.REZ – файла результатов расчета по программе SETD.FOR.

В качестве примера используем сетевой график, представленный на рис. 5.14.

Ниже приводятся программа расчета сетевого графика на языке FORTRAN и файл результатов расчета SETD.REZ.

```
100  Format (
      /' ***** ' /
      /' Расчет сетевого графика ' /
      /' Входные параметры: ' /
      /' M,N - число событий и работ в сети ' /
      /' II, JJ - номер события, из которого выходит ' /
      /' и в который входит работа ' /
      /' T - продолжительность выполнения I-й работы ' /)
```

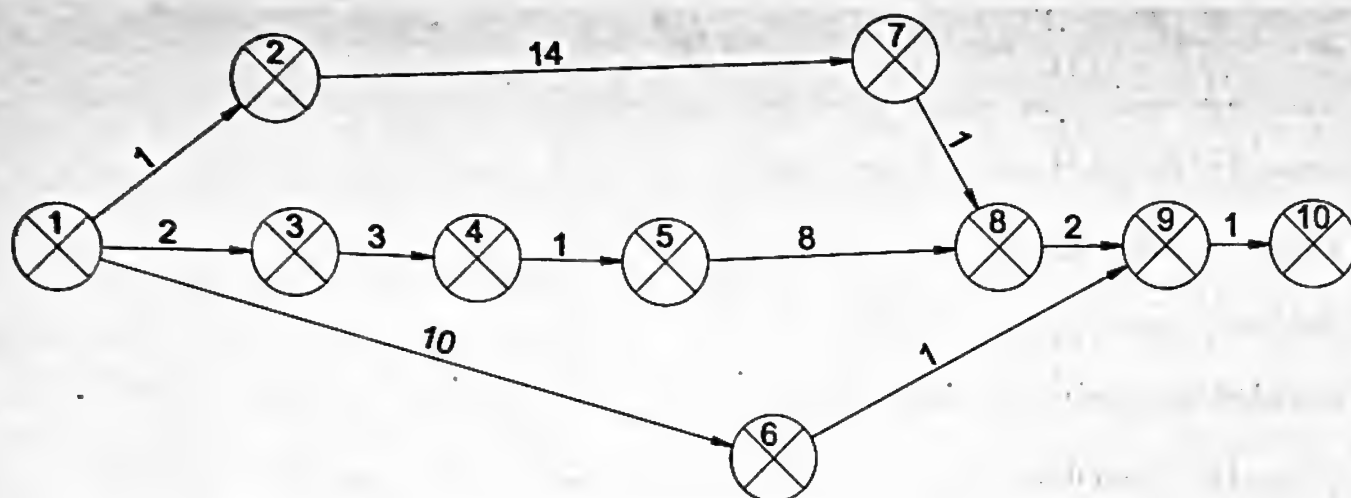


Рис. 5.14. Сетевой график выполнения проекта

```

101  Format (
      /' Выходные параметры:
      /' I-J - работа, выходящая из I-го
      /' события и входящая в J-е событие
      /' T - продолжительность выполнения I-J-й работы
      /' RII - полный резерв времени работы
      /' RC - свободный резерв времени работы
      /' TIIH - поздний срок начала работы
      /' TPO - ранний срок окончания работы
      /' TP, TII - ранний и поздний сроки
      /' наступления J-го события
      /' R - полный резерв времени события
      /' KP - события, лежащие на критическом пути
      DIMENSION KP(99),RII(99),II(99),JJ(99),T(99),
*   RC(99),TP(99),TII(99),R(99),X(99),L(99),TPO(99),TIIH(99)
      OPEN(5,FILE='SETD.DAN',STATUS='OLD')
      OPEN(6,FILE='SETD.REZ',STATUS='New')
      WRITE (6,100)
      WRITE (6,101)
      READ (5,20) M,N
      WRITE (6,20) M,N
20   FORMAT (2I5,F5.1)
      DO 1 I=1,N
      READ (5,20) II(I),JJ(I),T(I)
      WRITE (6,20) II(I),JJ(I),T(I)
21   FORMAT (2I5,6F8.1)
      1   L(I)=0
      TP(1)=0.
      DO 4 J=2,M
      TMAX=0.
      DO 5 K=1,N

```

```
IF (JJ(K).NE.J) GO TO 5
TP(J)=TP(II(K))+T(K)
IF (TP(J).LE.TMAX) GO TO 5
TMAX=TP(J)
5 CONTINUE
TP(J)=TMAX
4 CONTINUE
TII(M)=TP(M)
DO 6 IP=2,M
I=M-IP+1
TMIN=100000.
DO 7 KF=1,N
K=N-KF+1
IF (II(K).NE.I) GO TO 7
TII(I)=TII(JJ(K))-T(K)
IF (TII(I).GE.TMIN) GO TO 7
TMIN=TII(I)
7 CONTINUE
TII(I)=TMIN
6 CONTINUE
DO 8 J=1,M
8 R(J)=TII(J)-TP(J)
WRITE (6,22)
22 FORMAT (' I J T RII RC TIIH TPO '/')
DO 10 K=1,N
I=II(K)
J=JJ(K)
TPO(K)=TP(I)+T(K)
TIIH(K)=TII(J)-T(K)
RII(K)=TII(J)-TP(I)-T(K)
RC(K)=TP(J)-TP(I)-T(K)
WRITE (6,21) II(K),JJ(K),T(K),RII(K),RC(K),
* TIIH(K),TPO(K)
IF (ABS(R(I)-R(M)).GT..0001) GO TO 10
IF (ABS(R(J)-R(M)).GT..0001) GO TO 10
IF (ABS(R(I)-R(J)).GT..0001.OR.
* ABS(TIIH(K)-TII(I)).GT..0001) GO TO 10
L(K)=1
10 CONTINUE
WRITE (6,23)
23 FORMAT (' J TP TII R '/')
DO 9 J=1,M
9 WRITE (6,24) J,TP(J),TII(J),R(J)
24 FORMAT (4X,I5,3F8.1)
```



```

      KP(1)=1
      KK=1
      DO 11 K=1,N
      I=II(K)
      J=JJ(K)
      IF (L(K).EQ.0) GO TO 11
      IF (I.NE.KP(KK)) GO TO 11
      KK=KK+1
      KP(KK)=J
11    CONTINUE
      WRITE (6,25)
25    FORMAT (' Критический путь' /)
      WRITE (6,26) (KP(I),I=1,KK)
26    FORMAT (15I4)
      STOP
      END

```

Результаты расчета сохраняются в файле SETD.REZ в следующем виде:

Расчет сетевого графика

Входные параметры:

M,N - число событий и работ в сети

II, JJ - номер события, из которого выходит
и в который входит работа

T - продолжительность выполнения I-й работы

Выходные параметры:

I-J - работа, выходящая из I-го
события и входящая в J-е событие

T - продолжительность выполнения I-J-й работы

RII - полный резерв времени работы

RC - свободный резерв времени работы

TIIN - поздний срок начала работы

TPO - ранний срок окончания работы

TP, TII - ранний и поздний сроки
наступления J-го события

R - полный резерв времени события

KP - события, лежащие на критическом пути

10 11

1 2 1.0

1 3 2.0

1 6 10.0

2 7 14.0

3 4 3.0

4 5 1.0

5 8 8.0

6 9 1.0

7 8 1.0

8 9 2.0

9 10 1.0

I J T RII RC TIIH TPO

1 2 1.0 0.0 0.0 0.0 1.0

1 3 2.0 2.0 0.0 2.0 2.0

1 6 10.0 7.0 0.0 7.0 10.0

2 7 14.0 0.0 0.0 1.0 15.0

3 4 3.0 2.0 0.0 4.0 5.0

4 5 1.0 2.0 0.0 7.0 6.0

5 8 8.0 2.0 2.0 8.0 14.0

6 9 1.0 7.0 7.0 17.0 11.0

7 8 1.0 0.0 0.0 15.0 16.0

8 9 2.0 0.0 0.0 16.0 18.0

9 10 1.0 0.0 0.0 18.0 19.0

J TP TII R

1 0.0 0.0 0.0

2 1.0 1.0 0.0

3 2.0 4.0 2.0

4 5.0 7.0 2.0

5 6.0 8.0 2.0

6 10.0 17.0 7.0

7 15.0 15.0 0.0

8 16.0 16.0 0.0

9 18.0 18.0 0.0

10 19.0 19.0 0.0

Критический путь

1 2 7 8 9 10

Для создания графического изображения сетевого графика и результатов расчета необходимо определить размеры рисунка и кружочка. Всю эту информацию следует вводить в диалоговом режиме. Высота текста может быть определена, например, как 0.3 от радиуса кружочка. Если информация не вводится, она принимается по умолчанию. Как правило, информация по умолчанию указывается в угловых скобках, например, <20>.

Определить размеры рисунка и кружочка, а также высоту цифр на сетевом графике можно следующим образом:

```
(SETVAR "CMDECHO" 1) ; установка режима диалога
(SETQ BH (GETPOINT "\n Введите размер рисунка, <280,210>: ")
R (GETREAL "\n Введите радиус кружочка, <10>: "))
(SETVAR "CMDECHO" 0) ; отключение эха команд
```

```

(IF (EQ BH NIL) (SETQ BH (LIST 280 210))) ; размер рисунка
(IF (EQ R NIL) (SETQ R 10))               ; радиус кружочка по
                                           ; умолчанию
(SETQ H (* R 0.3))                        ; высота цифр на сетевом
                                           ; графике

```

Следующий шаг – считывание необходимой информации из файла результатов расчета. Допустим, файл результатов расчета SETD.REZ находится на диске С: в папке (каталоге) ORGANIZA. Открываем файл результатов расчета:

```
(SETQ F (OPEN "C:\\ORGANIZA\\SETD.REZ" "r"))
```

Считаем строки и сформируем из них списки. При этом строки (записи) с поясняющей информацией, а также пустые игнорируем.

; Функция (E) для считывания строки и представления ее в виде списка:

```

(DEFUN E ()
  (STRCAT " ( " (READ-LINE F) " ) ")
)

```

; Функция (EE) для пропуска поясняющих и пустых строк в файле
; результатов расчета SETD.REZ:

```

(DEFUN EE ()
  (WHILE (NOT (NUMBERP (CAR (SETQ SS (READ (E)))))) (CHR 10))
)

```

Считывание первой строки данных производится следующим образом:

```

(EE) ; пропуск поясняющих и пустых строк
(SETQ NNR SS ; считывание первой строки файла
           ; SETD.REZ
      NN (NTH 0 NNR)) ; определение числа событий
(PRINC (STRCAT " Число событий в сети - " (ITOA NN)))

```

Как уже отмечалось, кружочки на сетевом графике разделены на сектора с указанием номера события.

Для создания графического изображения кружочка с секторами разработаем функцию (SOBN...):

```

(DEFUN SOBN (N) ; функция изображения кружочка
  (SETVAR "CMDECHO" 0) ; с разделением на сектора
  (SETQ TN (ITOA N)
        TT (LIST (- (CAR TR) H) (+ (CAR (CDR TR)) H))
  )
  (COMMAND "CIRCLE" TR R
    "LINE" TR (POLAR TR (* PI 0.25) R) ""
    "LINE" TR (POLAR TR (* PI 0.75) R) ""
    "LINE" TR (POLAR TR (* PI 1.25) R) ""
    "LINE" TR (POLAR TR (* PI 1.75) R) ""
  )

```

"TEXT" TT H "0" TN) ; ввод номера события

Местоположение событий согласно логике выполнения работ и наглядности их представления пользователь сможет определить путем ввода соответствующих координат X и Y или с помощью мышки.

```
(SETQ K 0 LS '())
(REPEAT NN ; цикл ввода координат событий
  (SETQ K (+ 1 K))
  (SETVAR "CMDECHO" 1)
  (SETQ TR (GETPOINT (STRCAT "\n Введите координаты "
    (ITOA K) "-го события X,Y [можно мышкой] ")))
  (SETQ LS (CONS (LIST K TR) LS)) ; список координат событий
  (SOBN K) ; отображение события на экране
)
```

Ввод данных по работам целесообразно оформить в виде цикла:

```
(SETQ NR (NTH 1 NNR) ; определение числа работ в сети
  LTR '())
(REPEAT NR ; цикл изображения работ на экране
  (EE)
  (SETQ TIJ SS) ; считывание данных по работе
  (SETVAR "CMDECHO" 0) ; отключение эха команд
  (SETQ II (FIX (NTH 0 TIJ)) ; номер предшествующего события
    JJ (FIX (NTH 1 TIJ)) ; номер последующего события
    TRI (NTH 1 (ASSOC II LS)) ; точка начала работы
    TRJ (NTH 1 (ASSOC JJ LS)) ; точка окончания работы
    U (ANGLE TRI TRJ) ; угол наклона работы в радианах
    UGR (* U (/ 180 PI)) ; угол наклона работы в градусах
    D (DISTANCE TRI TRJ) ; длина стрелки (работы)
    TTT (POLAR TRI U (* D 0.4))
    TRI (POLAR TRI U R)
    TRJ (POLAR TRJ U (- 0 R))
    TRJ1 (POLAR TRJ U (- 0 (* 0.5 R)))
    TX (RTOS (NTH 2 TIJ) 2 0) ; длительность работы
    LE (LIST II JJ UGR TRI TRJ TTT) ; данные изображения работы
    LTR (CONS LE LTR) ; список данных для
    ; изображения работ
  (COMMAND "LINE" TRI TRJ1 "" ; изображение работы
    "PLINE" TRJ1 "W" (/ R 5) (/ R 25) TRJ "" ; изображение стрелки
    "TEXT" TTT H UGR TX) ; изображение длительности
    ; работы
)
```

Далее введем результаты расчета по работам и событиям, а также выполним их графическое изображение на сетевом графике:

```

(SETVAR "CMDECHO" 0)           ; отмена режима диалога
(SETQ LR '() LCC '())
(REPEAT NR                     ; цикл ввода из файла результатов
  (EE)                         ; расчета по работам
  (SETQ LR (CONS SS LR))
)
(SETQ LR (REVERSE LR))
(REPEAT NN                     ; цикл ввода из файла результатов
  (EE)                         ; расчета по событиям
  (SETQ LCC (CONS SS LCC))
)
(SETQ LCC (REVERSE LCC))
(SETQ M 0)
(FOREACH LC LCC                ; цикл изображения на графике
  ; результатов
  (SETQ M (+ 1 M)              ; расчета по событиям
    TR (NTH 1 (ASSOC M LS))
    TTP (POLAR TR (* PI 1.1) (* R 0.8)) ; точка изображения Тр
    TTII (POLAR TR (* PI 1.8) (* R 0.4)) ; точка изображения Тп
    TRP (POLAR TR (* PI 1.4) (* R 0.6))) ; точка изображения Rp
    (COMMAND "TEXT" TTP H "0" (RTOS (NTH 2 LC) 2 0)
      "TEXT" TTII H "0" (RTOS (NTH 1 LC) 2 0)
      "TEXT" TRP H "0" (RTOS (NTH 3 LC) 2 0))
  )
; цикл изображения результатов расчета по работам
(FOREACH EL LR
  (SETQ I (NTH 0 EL)           ; начальный индекс работы (I - J)
    J (NTH 1 EL)               ; конечный индекс работы (I - J)
    RII (NTH 3 EL)              ; полный резерв времени работы Rp
    RC (NTH 4 EL)               ; свободный резерв времени работы Rc
    TIIH (NTH 5 EL)             ; поздний срок начала работы Тпн
    TPO (NTH 6 EL)              ; ранний срок окончания работы Тпо
  )
; цикл изображения результатов расчета по работе
(FOREACH LE LTR
  (SETQ II (NTH 0 LE)           ; начальный индекс работы (I - J)
    JJ (NTH 1 LE)               ; конечный индекс работы (I - J)
    U (NTH 2 LE)                 ; угол наклона работы (I - J)
    TRI (NTH 3 LE)               ; точка начала работы (I - J)
    TRJ (NTH 4 LE)               ; точка конца работы
    TTT (NTH 5 LE)               ; точка изображения длительности
    ; работы
    TTIH (RTOS TIIH 2 0)         ; поздний срок начала работы Тпн

```



```

TPO (RTOS TPO 2 0) ; ранний срок окончания работы Тпо
TRII (RTOS RII 2 0) ; полный резерв времени работы Рп
TRC (RTOS RC 2 0) ; свободный резерв времени работы Rc
TR (STRCAT TRII "/" TRC) ; изображение резервов Рп/Rc
)
(IF (AND (EQ I II) (EQ J JJ))
  (COMMAND
    "TEXT" TRI H U TTIIH ; изображение Тпн
    "TEXT" (POLAR TRJ PI (* 2 H)) H U TPO ; изображение Тро
    "TEXT" (POLAR TTT (* PI 1.5) (* 1.5 H)) H U TR))) ; Рп/Rc
)

```

Теперь введем информацию по критическому пути и изобразим критический путь на графике утолщенной линией:

```

(EE) ; считывание из файла результатов расчета критического пути
(SETQ LK SS
  N 1)
(Foreach EK LK ; цикл изображения критического
  (SETQ L1 (LIST EK (NTH N LK))) ; пути утолщенной линией
  (SETQ N (+ 1 N))
  (Foreach LE LTR
    (SETQ II (NTH 0 LE) ; начальный индекс работы (I - J)
      JJ (NTH 1 LE) ; конечный индекс работы (I - J)
      TRI (NTH 3 LE) ; точка начала работы (I - J)
      TRJ (NTH 4 LE) ; точка конца работы (I - J)
      L2 (LIST II JJ) ; список работ (I - J)
    (IF (EQUAL L1 L2) ; условие изображения критического пути
      (COMMAND "PLINE" TRI "W" (/ R 10) (/ R 10) TRJ "))))
)

```

5.2.5. Комплекс функций создания изображения сетевого графика по результатам расчета его на языке Fortran

```

; *****
; Комплекс функций для создания изображения сетевого графика по
; результатам расчета его на языке Fortran
; *****
(SETVAR "CMDECHO" 1) ; установка режима диалога
(SETQ BH (GETPOINT "\n Введите размер рисунка, <280,210>: ")
  R (GETREAL "\n Введите радиус кружочка, <10>: "))
(SETVAR "CMDECHO" 0) ; отключение эха команд
(IF (EQ BH NIL) (SETQ BH (LIST 280 210))) ; размер рисунка

```

```

(IF (EQ R NIL) (SETQ R 10))      ; радиус кружочка по умолчанию
(SETQ H (* R 0.3))              ; высота цифр на сетевом графике
; Пример обращения к файлу результатов расчета показан ниже
(SETQ F (OPEN "C:\\\\ORGANIZA\\SETD.REZ" "r"))
; Создание функции (E) для считывания строки в виде списка
(DEFUN E () (STRCAT " ( " (READ-LINE F) " )"))
; Создание функции (EE) для пропуска поясняющих и пустых
; строк в файле результатов расчета SETD.REZ
(DEFUN EE ()
  (WHILE (NOT (NUMBERP (CAR (SETQ SS (READ (E)))))) (CHR 10))
)
(EE)                             ; пропуск поясняющих и пустых строк
(SETQ NNR SS                     ; считывание первой строки файла SETD.REZ
  NN (NTH 0 NNR))               ; определение числа событий
(PRINC (STRCAT " Число событий в сети - " (ITOA NN)))
(DEFUN SOBN (N)                 ; функция изображения события
  (SETVAR "CMDECHO" 0)
  (SETQ TN (ITOA N)
    TT (LIST (- (NTH 0 TR) (* 0.9 H)) (+ (NTH 1 TR) (* 1.1 H))))
  (COMMAND "CIRCLE" TR R)       ; отображение события кружочком
  (COMMAND                       ; разделение кружочка на сектора
    "LINE" TR (POLAR TR (* PI 0.25) R) ""
    "LINE" TR (POLAR TR (* PI 0.75) R) ""
    "LINE" TR (POLAR TR (* PI 1.25) R) ""
    "LINE" TR (POLAR TR (* PI 1.75) R) ""
    "TEXT" TT H "0" TN )       ; отображение номера события
  )
(COMMAND "LIMITS" "0,0" BH "ZOOM" "A")
(SETVAR "BLIPMODE" 0)          ; отключение маркеров
(SETQ K 0 LS '())
(REPEAT NN                      ; цикл ввода координат событий
  (SETQ K (+ 1 K))
  (SETVAR "CMDECHO" 1)
  (SETQ TR (GETPOINT (STRCAT "\n Введите координаты "
    (ITOA K) "-го события X,Y [можно мышкой] ")))
  (SETQ LS (CONS (LIST K TR) LS)) ; список координат событий
  (SOBN K)                       ; отображение события на экране
)
(SETQ NR (NTH 1 NNR)           ; определение числа работ в сети
  LTR '())
(REPEAT NR                      ; цикл отображения работ на экране
  (EE)
  (SETQ TIJ SS)                 ; считывание данных по работе

```

```

(SETVAR "CMDECHO" 0) ; отключение эха команд
(SETQ II (FIX (NTH 0 TIJ)) ; номер предшествующего события
JJ (FIX (NTH 1 TIJ)) ; номер последующего события
TRI (NTH 1 (ASSOC II LS)) ; точка начала работы
TRJ (NTH 1 (ASSOC JJ LS)) ; точка конца работы
U (ANGLE TRI TRJ) ; угол наклона работы в радианах
UGR (* U (/ 180 PI)) ; угол наклона работы в градусах
D (DISTANCE TRI TRJ) ; длина стрелки (работы)
TTT (POLAR TRI U (* D 0.4))
TRI (POLAR TRI U R)
TRJ (POLAR TRJ U (- 0 R))
TRJ1 (POLAR TRJ U (- 0 (* 0.5 R)))
TX (RTOS (NTH 2 TIJ) 2 0) ; длительность работы
LE (LIST II JJ UGR TRI TRJ TTT) ; данные, изображающие работу
LTR (CONS LE LTR) ; данные для изображения работ
(COMMAND "LINE" TRI TRJ1 "" ; отображение работы
"PLINE" TRJ1 "W" (/ R 5) (/ R 25) TRJ "" ; изображение стрелки
"TEXT" TTT H UGR TX) ; изображение длительности работы
)
(SETVAR "CMDECHO" 0) ; отмена режима диалога
(SETQ LR '() LCC '())
(REPEAT NR ; цикл ввода из файла результатов
(E) ; расчета по работам
(SETQ LR (CONS SS LR))
)
(SETQ LR (REVERSE LR))
(REPEAT NN ; цикл ввода из файла результатов
(E) ; расчета по событиям
(SETQ LCC (CONS SS LCC))
(SETQ LCC (REVERSE LCC)
M 0)
(Foreach LC LCC ; цикл отображения результатов
(SETQ M (+ 1 M) ; расчета по событиям
TR (NTH 1 (ASSOC M LS))
TTP (POLAR TR (* PI 1.1) (* R 0.8)) ; точка изображения Тр
TTII (POLAR TR (* PI 1.8) (* R 0.4)) ; точка изображения Тп
TRP (POLAR TR (* PI 1.4) (* R 0.6)) ; точка изображения Рп
(COMMAND "TEXT" TTP H "0" (RTOS (NTH 2 LC) 2 0)
"TEXT" TTII H "0" (RTOS (NTH 1 LC) 2 0)
"TEXT" TRP H "0" (RTOS (NTH 3 LC) 2 0))
)
(Foreach EL LR ; цикл ввода результатов расчета по работам
(SETQ I (NTH 0 EL) ; начальный индекс работы (I - J)
J (NTH 1 EL) ; конечный индекс работы (I - J)

```

```

    RII (NTH 3 EL)           ; полный резерв времени работы Rп
    RC (NTH 4 EL)           ; свободный резерв времени работы Rc
    TIIH (NTH 5 EL)         ; поздний срок начала работы Tпн
    TPO (NTH 6 EL)          ; ранний срок окончания работы Тро
)
(Foreach LE LTR             ; цикл ввода результатов расчета по работе
  (SETQ II (NTH 0 LE)       ; начальный индекс работы (I - J)
    JJ (NTH 1 LE)          ; конечный индекс работы (I - J)
    U (NTH 2 LE)           ; угол наклона работы (I - J)
    TRI (NTH 3 LE)         ; точка начала работы (I - J)
    TRJ (NTH 4 LE)         ; точка конца работы
    TTT (NTH 5 LE)         ; точка отображения длительности работы
    TTIH (RTOS TIIH 2 0)   ; поздний срок начала работы Tпн
    TTPO (RTOS TPO 2 0)    ; ранний срок окончания работы Тро
    TRII (RTOS RII 2 0)     ; полный резерв времени работы Rп
    TRC (RTOS RC 2 0)       ; свободный резерв времени работы Rc
    TR (STRCAT TRII "/" TRC) ; отображение резервов в виде Rп/Rc
  )
  (IF (AND (EQ I II) (EQ J JJ))
    (COMMAND
      "TEXT" TRI H U TTIH           ; отображение Tпн
      "TEXT" (POLAR TRJ PI (* 2 H)) H U TTPO ; отображение Тро
      "TEXT" (POLAR TTT (* PI 1.5) (* 1.5 H)) H U TR))) ; Rп/Rc
  )
)
(EE)
(SETQ LK SS                 ; считывание из файла результатов
  N 1)                     ; расчета критического пути
(Foreach EK LK              ; цикл отображения критического
  (SETQ L1 (LIST EK (NTH N LK)) ; пути утолщенной линией
    N (+ 1 N))
  (Foreach LE LTR
    (SETQ II (NTH 0 LE)       ; начальный индекс работы (I - J)
      JJ (NTH 1 LE)          ; конечный индекс работы (I - J)
      TRI (NTH 3 LE)         ; точка начала работы (I - J)
      TRJ (NTH 4 LE)         ; точка окончания работы (I - J)
      L2 (LIST II JJ))       ; список работ (I - J)
    (IF (EQUAL L1 L2)         ; условие отображения критического пути
      (COMMAND "PLINE" TRI "W" (/ R 10) (/ R 10) TRJ "")))
  )
)
(CLOSE F)

```

После загрузки и выполнения комплекса функций из файла SET-DFOR.LSP на экране будет формироваться изображение сетевого графика. В окончательном виде сетевой график выполнения проекта представлен на рис. 5.15.

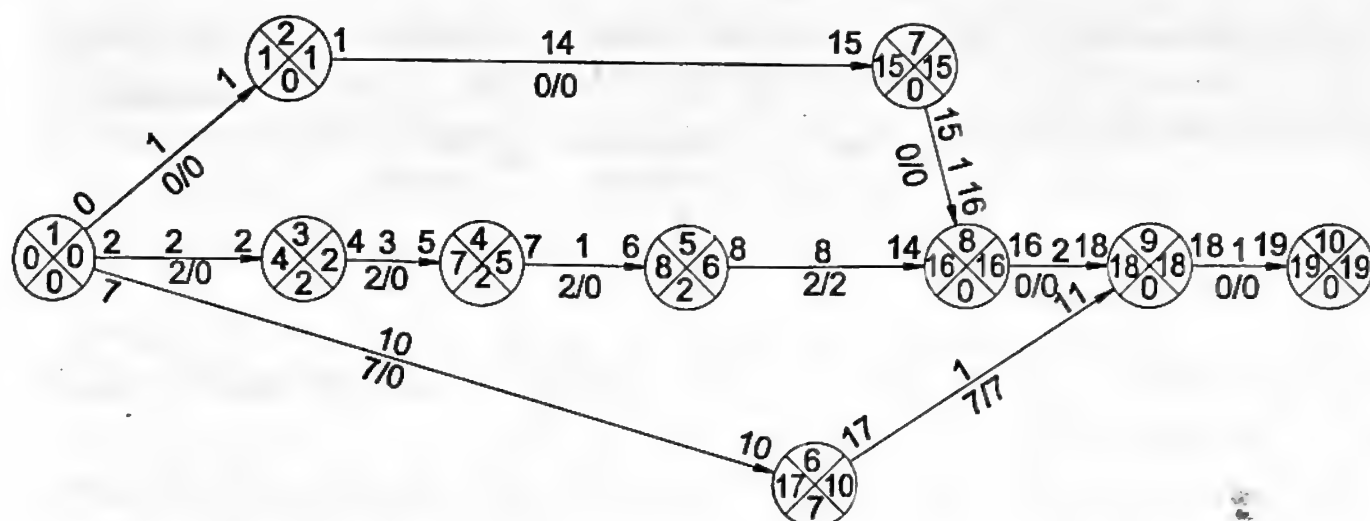


Рис. 5.15. Сетевой график выполнения проекта

5.3. Разработка экспертной системы

Экспертные системы относятся к числу интеллектуальных вычислительных систем и предназначены для моделирования или имитации поведения опытных специалистов-экспертов при решении какого-либо узкого вопроса.

Экспертные системы позволяют накапливать, систематизировать и сохранять знания и профессиональный опыт специалистов высокой квалификации, занятых в первую очередь в тех областях, где задачи и их решения формализованы слабо или совсем неформализованы (неструктурированы), например, в машиностроении, экономике, вычислительной технике, медицине и др.

Таким образом, *экспертная система* представляет собой программный комплекс, обеспечивающий накопление, хранение, обновление и возможность использования специально подготовленных композиций знаний.

Структурно экспертная система состоит из трех основных частей: интерфейса, механизма вывода и базы знаний (рис. 5.16).

Интерфейс – это некоторая подсистема, язык взаимодействия пользователя с машиной, экспертной системой на ограниченном естественном языке или на специфическом искусственном языке, используемом в данной предметной области.

Механизм вывода представляет собой комплекс программ, который использует базу знаний, факты, гипотезы, учитывает запросы пользователя и тем самым позволяет решать поставленные задачи. В механизмах вывода используются несколько стратегий вывода (то есть переходов от одного правила к другому) – прямой, обратный и комбинированный.

База знаний – это совокупность фактов и знаний из той или иной области, определенным образом представленных в программном виде пользователями, экспертами и инженерами знаний. База знаний содержит два

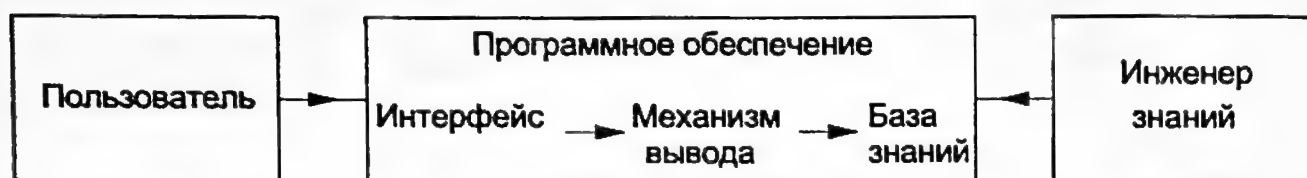


Рис. 5.16. Упрощенная схема функционирования экспертной системы

основных элемента: факты (данные) из предметной области и специальные эвристики или правила, которые управляют использованием фактов при решении проблем.

Качество заключений, выводов, рекомендаций, получаемых с помощью той или иной экспертной системы, определяется по способу представления знаний, объему базы знаний и мощности используемой машины вывода.

Сложность и многообразие структур знаний вызвали к жизни несколько различных способов их представления, из которых следует выделить логическое исчисление, фреймовые и продукционные системы, семантические сети. Каждый способ представления знаний имеет свои достоинства и недостатки.

Логическое исчисление основано на представлении знаний с помощью предикатов и применении к ним формальных методов вывода, например, метода резолюций и др. Предикат представляет собой отношение между объектами. Например, предикат *Брат (Петя, Витя)* означает, что *Петя* – брат *Вити*. В скобках указаны объекты, к которым применяется данный предикат (отношение) *Брат*.

Такая форма представления знаний нашла широкое применение в языке логического программирования PROLOG.

Фреймовая система основана на представлении знаний в виде определенных структур, называемых фреймами.

Основу *семантических сетей* составляет представление знаний в виде ориентированного графа с размеченными вершинами и дугами. Вершинам соответствуют объекты или ситуации, а дугам – отношения между ними (рис. 5.17).

Представление знаний в такой форме более наглядно и понятно. Семантические сети адекватно отражают взаимосвязь объектов.

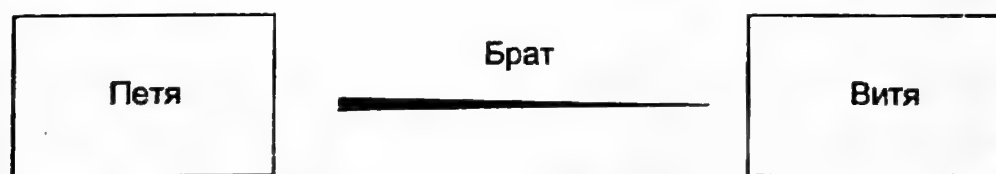


Рис. 5.17. Схема представления знаний в семантической сети

Продукционные системы основаны на представлении знаний в виде совокупности условий и выводов. Если выполняются определенные условия, то будет такой-то результат, вывод. Например, если *Петя* – сын дяди *Миши* и *Витя* – сын того же дяди *Миши*, то *Петя* – брат *Вити*.

Рассмотрим основные этапы разработки простейшей экспертной системы.

5.3.1. Постановка задачи

Следует разработать комплекс функций AutoLISP простейшей экспертной системы. Представление знаний обеспечивается с помощью продукционной системы, использующей представление знаний (правил) в виде: *если – то*.

5.3.2. Выявление основных особенностей, взаимосвязей и количественных закономерностей

В качестве стратегий вывода результата используют прямой, обратный и комбинированный выводы (стратегии).

При *прямом* выводе отправной точкой служит известный начальный набор фактов. С учетом его и базы правил методом прямого сравнения имеющихся фактов и условий осуществляется поиск результата.

При *обратном* выводе отправным моментом служит список предполагаемых результатов (выводов, заключений, действий), которым в конечном счете ставится в соответствие множество фактов (условий), делающих их правомочными.

Совместное их использование возможно путем определенного чередования.

Рассмотрим стратегию прямого вывода. Имеются несколько фактов. Обозначим их F1, F2, F3, F5, F7, F8.

Допустим, существует несколько правил. Представим их, как на рис. 5.18.

Эти правила представляют базу знаний. Чтобы найти результат F9, механизм вывода сравнивает начальный набор фактов (F1 F2 F3 F5 F7 F8) с фактами, содержащимися в базе знаний в правилах, изложенных в виде

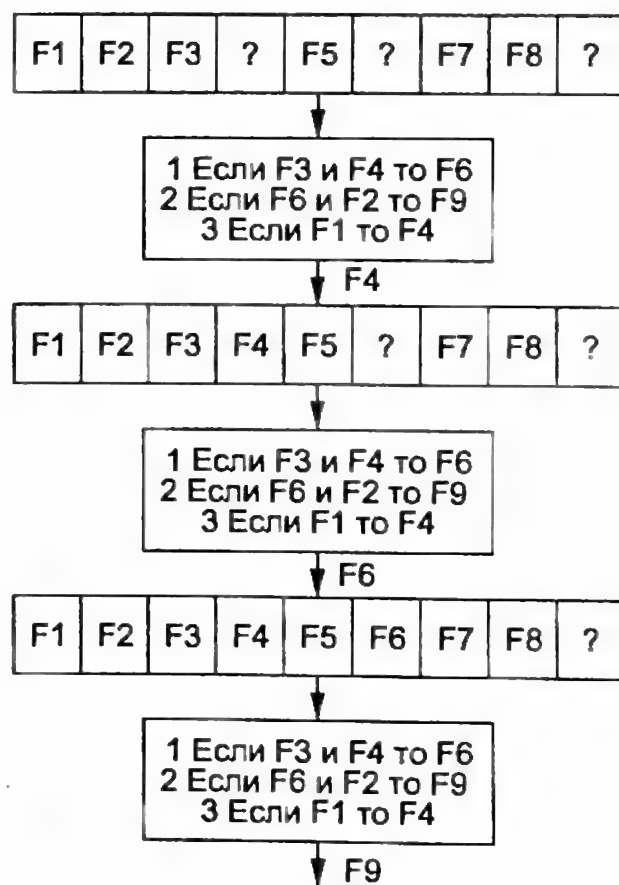


Рис. 5.18. Правила выбора результата

условий. Используя правило под номером 3 в базе знаний, список фактов можно расширить.

Правило 3 гласит: если мы имеем факт F1, получим вывод (результат) F4, который вносится в расширенный список фактов. Механизм вывода снова сравнивает, но уже расширенный список фактов (F1 F2 F3 F4 F5 F7 F8) с фактами, содержащимися в базе знаний в правилах, представленных в виде условий. В данном случае согласно правилу под номером 1 в список фактов добавляется факт F6. На следующем этапе в соответствии с правилом 2 получим искомый результат – факт F9.

5.3.3. Разработка последовательности действий и комплекса функций для создания экспертной системы

Рассмотрим простейшую экспертную систему, с помощью которой по имеющимся фактам можно определить неисправности в двигателе.

Создадим тестовый файл EKSPERT.DAN, в котором собраны известные начальные факты о неисправностях в двигателе. Допустим, этот файл находится на диске C: в папке (каталоге) VLAutoLISP и в файле имеются следующие факты, которые вводятся как строковые константы с начальной пустой строкой:

```
"Слабая искра на свечах"
"Слабо вращается коленвал"
*
```

Для ввода фактов в виде списка в экспертную систему откроем файл данных (фактов) и создадим на AutoLISP функцию (**READF...**) для считывания начальных фактов, вывода их на экран и в файл результатов:

```
; Открытие файла данных EKSPERT.DAN для считывания фактов
(SETQ FD (OPEN "C:\\VLAutoLISP\\EKSPERT.DAN" "r"))
; Создание функции для считывания строки в виде списка
(DEFUN READL (FF)
  (STRCAT "( " (READ-LINE FF) " )"))
; Открытие файла EKSPERT.DOC для записи фактов и результатов
(SETQ FW (OPEN "C:\\VLAutoLISP\\EKSPERT.DOC" "a"))
(DEFUN READF ( / Z) ; считывание фактов из файла данных
  (SETQ FAKTI '())
  (WHILE (NOT (EQ (SETQ Z (READ (READL FD))) NIL))
    (SETQ FAKTI (CONS Z FAKTI)))
  (PRINC "\n " FW)
  (PRINC "\n ")
  (PRINC FAKTI)
  (PRINC FAKTI FW)
)
```

Начальные факты можно ввести и в диалоговом режиме, используя функцию (**GETSTRING...**), которая запрашивает у пользователя строковую константу и возвращает ее.

```
(DEFUN READD ( / Z) ; считывание фактов в диалоговом режиме
  (SETQ FAKTI '())
  (PRINC "\n Введите факты в кавычках на отдельных строках")
  (PRINC "\n для завершения нажмите клавишу Enter") (PRINC "\n ")
  (WHILE (NOT (EQUAL (SETQ Z (READ (STRCAT "( " (READ-LINE)
    " )")))) NIL))
  (SETQ FAKTI (CONS Z FAKTI)))
  (PRINC "\n " FW)
  (PRINC FAKTI FW)
)
```

Далее создадим текстовый файл правил (знаний) EKSPERT.PRA, в котором будут находиться правила, сформулированные неким экспертом. Этот файл часто называют *базой знаний*. В нашем примере файл может выглядеть следующим образом (с пустой строкой в конце):

```
1 ("нет искры на свечах") ("двигатель не запускается")
2 ("плохое соединение проводов") ("малый ток в первичной кз")
3 ("малый ток в первичной кз") ("малый ток во вторичной кз")
4 ("малый ток во вторичной кз") ("слабая искра на свечах")
5 ("слабая искра на свечах") ("плохой аккумулятор")
6 ("плохой аккумулятор") ("малый ток в первичной кз")
7 ("слабо вращается колен вал") ("плохой аккумулятор")
8 ("пробит конденсатор") ("нет тока в первичной кз")
9 ("нет тока в первичной кз") ("нет тока во вторичной кз")
10 ("нет тока во вторичной кз") ("нет искры на свечах")
```

Первый элемент в строке указывает на номер правила, последний — на получаемый вывод (результат). Все элементы строки между первым и последним — это условия, при которых может быть получен результат. Так, правило под номером 1 означает следующее: если нет искры на свечах, двигатель не запускается. Здесь применена упрощенная схема представления правила.

Для ввода правил в виде списка в экспертную систему откроем файл правил (знаний) и создадим на AutoLISP функцию (**READP...**) для считывания базы знаний:

```
; Открытие файла знаний EKSPERT.PRA для считывания правил
(SETQ FP (OPEN "C:\\VLAutoLISP\\EKSPERT.PRA" "r"))
(DEFUN READP (/ Z) ; считывание правил из файла EKSPERT.PRA
  (SETQ BAZA '())
  (WHILE (NOT (EQ (SETQ Z (READ (READL FP))) NIL))
    (SETQ BAZA (CONS Z BAZA)))
  (SETQ BAZA (REVERSE BAZA)))
```

Для поиска искомого ответа по имеющимся фактам при наличии соответствующей базы знаний создадим на AutoLISP функцию (**EKSPERT...**), которая должна отражать начальные факты и результаты работы экспертной системы на экране и в файле EKSPERT.REZ:

```
(DEFUN EKSPERT (BAZA) ; функция поиска вывода
  (PRINC "\n\n К О М М Е Н Т А Р И Й: ")
  (PRINC "\n Результаты работы экспертной системы выводятся ")
  (PRINC "\n на экран и в файл EKSPERT.DOC")
  (PRINC "\n\n Начальные факты в файле EKSPERT.DAN: " FW)
  (PRINC "\n\n Начальные факты в файле EKSPERT.DAN: " )
  (READF) ; считывание из файла данных начальных фактов
  (READP) ; считывание из файла знаний правил
  (PRINC "\n\n Р Е З У Л Ь Т А Т Ы: ")
  (FOREACH PR BAZA ; цикл просмотра правила
    ; в базе знаний
    (SETQ USL (CDR (REVERSE (CDR PR)))) ; выделение условий
    (COND ((PODMNOJ USL FAKTI) ; если условие выполнено - поиск:
      (SETQ V (LAST PR) ; выделение вывода (результата)
        NP (CAR PR)) ; определение номера правила
      (PRINC (STRCAT "\n Согласно правилу " (ITOA NP) " из "
        "базы знаний EKSPERT.PRA ответ: ") FW)
      (PRINC (CAR V) FW) ; распечатка вывода в файл результатов
      (PRINC (STRCAT "\n Согласно правилу " (ITOA NP) " из "
        "базы знаний EKSPERT.PRA ответ: \n" ) )
      (PRINC (CAR V)) ; распечатка вывода на экран
      (SETQ FAKTI (CONS V FAKTI))) ; ввод вывода в список фактов
      (T FAKTI)))
    (PRINC "\n К О Н Е Ц! ") (PRIN1)
  )
)
```

В функции (**EKSPERT...**) используются две вспомогательные функции (**PERMN...**) и (**PODMNOJ...**) для определения пересечения двух множеств элементов в списках X, Y и проверки вхождения множества элементов списка X в множество элементов списка Y соответственно:

```
(DEFUN PERMN (X Y) ; функция пересечения множеств X и Y
  (COND ((NULL X) NIL)
    ((MEMBER (CAR X) Y) (CONS (CAR X) (PERMN (CDR X) Y)))
    (T (PERMN (CDR X) Y)))
)

(DEFUN PODMNOJ (X Y) ; функция проверки вхождения множества X в Y
  (EQUAL X (PERMN X Y))
)
```

В окончательном виде простейшая экспертная система со стратегией прямого вывода будет выглядеть следующим образом.

5.3.4. Комплекс функций для создания экспертной системы

```
; *****
; Простейшая экспертная система со стратегией прямого вывода
; *****
; Открытие файла данных EKSPERT.DAN для считывания фактов
(SETQ FD (OPEN "C:\\VLAUTOLISP\\EKSPERT.DAN" "r"))
; Открытие файла знаний EKSPERT.PRA для считывания правил
(SETQ FP (OPEN "C:\\VLAUTOLISP\\EKSPERT.PRA" "r"))
; Открытие файла EKSPERT.DOC для записи фактов и результатов
(SETQ FW (OPEN "C:\\VLAUTOLISP\\EKSPERT.DOC" "a"))
(TEXTSCR)
; Создание функции для считывания строки в виде списка
(DEFUN READL (FF) (STRCAT "( " (READ-LINE FF) " )"))
(DEFUN READF ( / Z) ; считывание фактов из файла данных
  (SETQ FAKTI '())
  (WHILE (NOT (EQ (SETQ Z (READ (READL FD))) NIL))
    (SETQ FAKTI (CONS Z FAKTI)))
  (PRINC "\n " FW) (PRINC "\n ") (PRINC FAKTI) (PRINC FAKTI FW)
)
(DEFUN READP ( / Z) ; считывание правил из файла EKSPERT1.PRA
  (SETQ BAZA '())
  (WHILE (NOT (EQ (SETQ Z (READ (READL FP))) NIL))
    (SETQ BAZA (CONS Z BAZA)))
  (SETQ BAZA (REVERSE BAZA))
)
(DEFUN PERMN (X Y) ; функция пересечения множеств X и Y
  (COND ((NULL X) NIL)
        ((MEMBER (CAR X) Y) (CONS (CAR X) (PERMN (CDR X) Y)))
        (T (PERMN (CDR X) Y)))
)
(DEFUN PODMNOJ (PODMN MNOJ)
  (EQUAL PODMN (PERMN PODMN MNOJ))
)
(DEFUN EKSPERT (BAZA) ; функция поиска вывода
  (PRINC "\n\n К О М М Е Н Т А Р И Й: ")
  (PRINC "\n Результаты работы экспертной системы выводятся ")
  (PRINC "\n на экран и в файл EKSPERT.DOC")
  (PRINC "\n\n Начальные факты в файле EKSPERT.DAN: " FW)
  (PRINC "\n\n Начальные факты в файле EKSPERT.DAN: " )
  (READF) ; считывание из файла данных
```

```

; начальных фактов
(READP) ; считывание из файла
; знаний правил
(PRINC "\n\n Р Е З У Л Ь Т А Т Ы: ")
(FOREACH PR BAZA ; цикл просмотра правила в базе знаний
  (SETQ USL (CDR (REVERSE (CDR PR)))) ; выделение условий
  (COND ((PODMNOJ USL FAKTI) ; если условие выполнено - поиск
    (SETQ V (LAST PR) ; выделение вывода (результата)
      NP (CAR PR)) ; определение номера правила
    (PRINC (STRCAT "\n Согласно правилу " (ITOA NP) " из "
      "базы знаний EKSPERT.PRA ответ: ") FW)
    (PRINC (CAR V) FW) ; распечатка вывода в файл
      ; результатов
    (PRINC (STRCAT "\n Согласно правилу " (ITOA NP) " из "
      "базы знаний EKSPERT.PRA ответ: \n" ) )
    (PRINC (CAR V)) ; распечатка вывода на экран
    (SETQ FAKTI (CONS V FAKTI ))) ; ввод вывода в список фактов
      (T FAKTI))
) (PRINC "\n К О Н Е Ц! ") (PRIN1)
)
(EKSPERT BAZA)
(CLOSE FD)
(CLOSE FP)
(CLOSE FW)

```

Результаты расчета представлены в окне консоли Visual LISP (рис. 5.19).

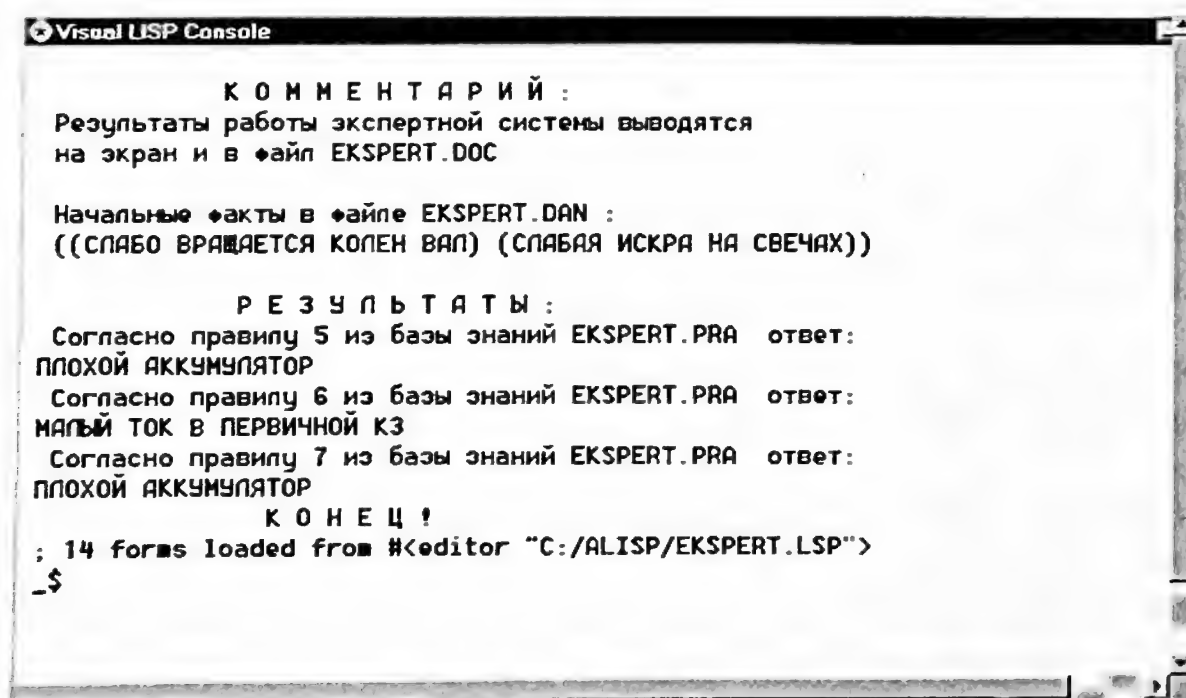


Рис. 5.19. Результат выполнения функции (EKSPERT...)

СОЗДАНИЕ ПОДСИСТЕМ ДЛЯ ВЫПОЛНЕНИЯ ИНЖЕНЕРНЫХ РАСЧЕТОВ

Расчет и представление динамических параметров	242
Расчет и создание параметрического изображения нагрузок в фермах.....	252
Расчет и параметрическое изображение различных способов выполнения производственного процесса ..	270
Оптимизация загрузки оборудования	288

Эта глава содержит примеры создания подсистем для выполнения инженерных расчетов: динамических характеристик силового агрегата; линий влияния; различных способов организации производственного процесса; определения загрузки оборудования.

6.1. Расчет и представление динамических параметров

6.1.1. Постановка задачи

Задан некоторый агрегат, состоящий из двух звеньев: ведущего звена (кривошип) и ведомого звена (поступательно движущаяся кулиса). Во время рабочего хода ведомое звено реализует постоянное усилие P . Вес ведомого звена G . Весом остальных звеньев агрегата и силами трения пренебрегаем. Момент инерции вращающихся частей ведущего звена равен J_1 . Для заданных параметров агрегата (рис. 6.1) – $P = 100$ Н, $G = 10$ Н, $J_1 = 0,05$ кгм², длина звена (кривошипа) $L = 0,1$ м – необходимо рассчитать и представить в графическом виде основные характеристики агрегата:

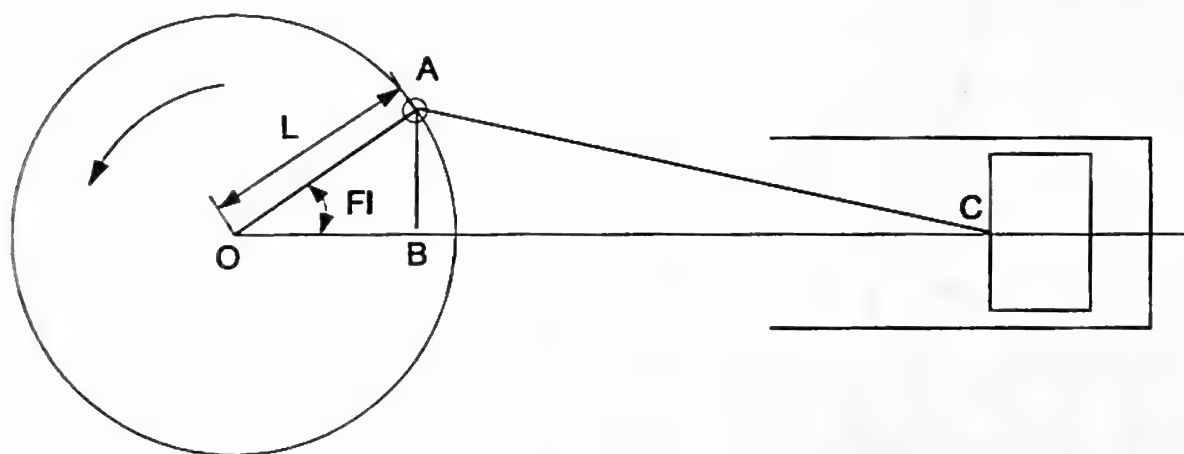


Рис. 6.1. Схема силового агрегата

- приведенный момент от движущих сил $M_{дл}$;
- приведенный момент от сил сопротивления $M_{ср}$;
- приведенный момент инерции $J_{1р}$;
- угловую скорость ведущего звена ω_1 ;
- полную кинетическую энергию агрегата E .

Необходимо определить основные характеристики агрегата для режима разгона, установившегося движения и торможения. При этом следует учитывать, что агрегат выходит на режим установившегося режима через два оборота ($N = 2$). Все основные характеристики необходимо определить в функции угла поворота ведущего звена – φ и представить их на экране дисплея графически во время работы агрегата. Предварительно предстоит разработать соответствующие функции на AutoLISP.

6.1.2. Выявление основных особенностей, взаимосвязей и количественных закономерностей

Искомые основные характеристики силового агрегата вначале определим для режима разгона.

Используя равенство мощностей на входе и выходе, вычислим приведенный момент от сил сопротивления $M_{\text{ср}}$:

$$M_{\text{ср}} \omega_1 = P V_{\text{с}},$$

где $V_{\text{с}}$ – линейная скорость ведомого звена.

Угловая скорость ведущего звена определяется по формуле:

$$\omega_1 = V_{\text{а}} / L,$$

где $V_{\text{а}}$ – линейная скорость точки А ведущего звена, направленная по касательной к окружности О.

Линейную скорость точки А можно разложить на две составляющие: по вертикали $V_{\text{ау}}$ и горизонтали $V_{\text{аг}}$:

$$V_{\text{аг}} = V_{\text{а}} \sin (180^\circ - \varphi) = V_{\text{а}} \sin \varphi$$

Следовательно, скорости точек А и С по горизонтали равны:

$$V_{\text{аг}} = V_{\text{с}}, V_{\text{с}} = V_{\text{а}} \sin \varphi = \omega_1 L \sin \varphi$$

Используя это равенство, выражение для приведенного момента от сил сопротивления можно представить в следующем виде:

$$M_{\text{ср}} \omega_1 = P \omega_1 L \sin \varphi, M_{\text{ср}} = P L \sin \varphi$$

Используя равенство работы движущего момента $A_{\text{д}}$ и сил сопротивления $A_{\text{с}}$ за один цикл, определим величину приведенного момента от движущих сил $M_{\text{дп}}$. При этом будем учитывать действие сил сопротивления в течение первой половины оборота ведущего звена, когда действует полезное сопротивление движению P .

$$A_{\text{д}} = \int_0^{2\pi} M_{\text{дп}} d\varphi; A_{\text{с}} = 2PL; \int_0^{2\pi} M_{\text{дп}} d\varphi = 2PL; M_{\text{дп}} 2\pi = 2PL$$

где $2L$ – расстояние перемещения ведомого звена за пол-оборота ведущего звена.

Следовательно,

$$M_{\text{дп}} = PL / \pi$$

Движущий момент мотора считается постоянным и действует во время разгона и установившегося движения.

Исходя из равенства кинетических энергий, приведенный момент инерции $J_{\text{ин}}$ к ведущему звену можно определить следующим образом:

$$1/2 J_{\text{ин}} \omega_1^2 = 1/2 J_1 \omega_1^2 + 1/2 G V_{\text{с}}^2 / g_{\text{у}},$$

откуда $J_{\text{ин}} = J_1 + G V_{\text{с}}^2 / (g_{\text{у}} \omega_1^2)$, но $V_{\text{с}} = \omega_1 L \sin \varphi$,

следовательно, $J_{\text{ин}} = J_1 + G L^2 \sin^2 \varphi / g_{\text{у}}$

Исходя из теоремы живых сил, определим значение угловой скорости ведущего звена:

$$1/2 J_{1n} \omega_1^2 = A_d,$$

$$\text{но } A_d = \int_0^\varphi M_{дп} d\varphi = \frac{PL\varphi}{\pi}, \text{ откуда } \omega_1 = \sqrt{\frac{2PL\varphi}{J_{1n} \pi}}$$

Рассмотрим теперь установившийся режим работы агрегата. Во время движения агрегата в первой половине оборота ведущего звена при изменении угла φ от 0 до π действуют приведенный движущий момент $M_{дп}$ и приведенный момент от сил сопротивления $M_{ср}$. Результирующий приведенный момент M_p в первой половине оборота звена 1 будет равен:

$$M_p = M_{дп} - M_{ср} = PL / \pi - PL \sin \varphi = PL (1 / \pi - \sin \varphi).$$

За это же время полная кинетическая энергия агрегата (в первой половине оборота ведущего звена) составит:

$$E = A_d - A_c + C,$$

$$E = \int_0^\varphi M_{дп} d\varphi - \int_0^\varphi M_{ср} d\varphi + C = \int_0^\varphi \frac{PL}{\pi} d\varphi - \int_0^\varphi PL \sin \varphi d\varphi$$

после интегрирования

$$E = PL (\varphi / \pi + \cos \varphi) + C,$$

где C – постоянная, которая определяется по начальным данным при $\varphi = 0$ и $E = E_0$.

$$C = E_0 - PL, \text{ так как } \cos 0 = 1,$$

при этом E_0 – кинетическая энергия, накопленная за время разгона.

Следовательно, кинетическая энергия агрегата составит:

$$E = PL (\varphi / \pi + \cos \varphi) + E_0 - PL$$

Накопленная кинетическая энергия за время разгона:

$$E_0 = \int_0^{N2\pi} M_{дп} d\varphi = \int_0^{N2\pi} \frac{PL}{\pi} d\varphi = 2PLN$$

Полную кинетическую энергию агрегата можно рассчитать по формуле:

$$E = PL (\varphi / \pi + \cos \varphi - 1 + 2N)$$

Полная кинетическая энергия агрегата во вторую половину оборота ведущего звена (от $\varphi = \pi$ до $\varphi = 2\pi$):

$$E = A_d - A_c + E_0 = PL \varphi / \pi - 2PL + 2PLN = PL (\varphi / \pi - 2 + 2N)$$

Во второй половине оборота ведущего звена холостой ход и $P = 0$, следовательно, работа сил сопротивления $A_c = -2PL$ останется неизменной.

Исходя из теоремы живых сил ($J_{1n} \omega_1^2 / 2 = E$), угловую скорость ведущего звена ω_1 при установившемся движении агрегата можно определить по формуле:

$$\omega_1 = \sqrt{\frac{2E}{J_{1n}}}$$

Далее рассмотрим режим остановки (торможения) агрегата. Определим приведенный момент сопротивления M_{cp} , который действует в первой половине оборота ведущего звена (от $\varphi = 0$ до $\varphi = \pi$):

$$M_{cp} = P L \sin \varphi \quad (0 \leq \varphi \leq \pi), \quad (2\pi \leq \varphi \leq 3\pi), \dots$$

Значение кинетической энергии агрегата за первую половину оборота ведущего звена (от $\varphi = 0$ до $\varphi = \pi$) составит:

$$E_1 = E_0 - A_c = 2 P L N - P L (1 - \cos \varphi) = P L (2N + \cos \varphi - 1)$$

За вторую половину оборота ведущего звена кинетическая энергия будет постоянна и равна кинетической энергии при $\varphi = \pi$:

$$E_2 = P L (2N + \cos \pi - 1) = P L (2N - 1 - 1) = 2 P L (N - 1)$$

При дальнейшем вращении ведущего звена от $\varphi = 2\pi$ до 3π кинетическую энергию можно вычислить по формуле:

$$E_3 = E_2 - P L (1 - \cos \varphi) = 2 P L (N - 1) - P L (1 - \cos \varphi) = P L (2N - 3 + \cos \varphi)$$

Изменение угловой скорости ведущего звена при торможении и остановке составит:

$$\omega_1 = \sqrt{\frac{2E}{J_{1n}}}$$

6.1.3. Разработка последовательности действий и комплекса функций расчета и параметрического изображения агрегата и всех его характеристик

Порядок расчета и графического представления результатов включает следующие основные этапы:

- *определение вспомогательных данных и начальных значений.* Введем начальный угол поворота ведущего звена $\varphi = 0$ и начальное перемещение на графике изображения той или иной характеристики $Z = 0$. Размер графика по оси X принят равным $4L$. Далее введем значение ускорения свободного падения $g_u = 9.81$ и определим приращение угла поворота ведущего звена (принято равным 30°). Исходя из заданного числа оборотов T ведущего звена для разгона, определяем число дискретных положений ведущего звена – NZ , для которых проводится расчет искомых параметров и выдается их изображение. Затем определим базовые точки для построения графиков;

- *представление названий и базовых линий для искомых параметров.* Базовые точки для базовых линий определяются со смещением относительно друг друга на расстояние $0.6 L$, а начальные точки для текста – со смещением относительно линий на расстояние $0.5 L$ над линией;
- *создание графического изображения агрегата, расчет и представление искомых параметров.* Для создания графического изображения агрегата предварительно определим угол поворота ведущего звена, расстояния OB , BA , координаты точек T_A , T_B , T_1 , T_2 , T_3 , которые функционально зависят от угла φ . После создания графического изображения агрегата с помощью полилинии его необходимо запомнить, с тем чтобы удалить после выполнения расчетов искомых параметров. Затем предстоит сформировать новое положение агрегата, рассчитать новые значения искомых параметров, построить их изображение и т. д. Расчет параметров осуществляется в цикле. Сначала рассчитывается приведенный момент от движущих сил $M_{дп}$, затем – работа от движущих сил $A_{дп}$, приведенный момент инерции $J_{ин}$ и угловая скорость ведущего звена. В конце периода разгона определяются и изображаются на графиках угловые значения искомых величин.

Разработка комплекса функции AutoLISP:

- *определение вспомогательных данных и начальных значений.* Эта часть программы может быть записана в следующем виде:

```
(SETQ GU 9.81 ; ускорение свободного падения
      I 1 ; номер исследуемого параметра
      DFI (/ PI 6) ; приращение угла поворота
      FI (- 0 DFI) ; угол поворота ведущего звена
      NZ (/ (* N 2 PI) DFI) ; число значений исследуемого параметра
      NZ (+ 1 (FIX NZ))
      DZ (/ (* 4 L) NZ) ; смещение по горизонтали в цикле
      Z (- 0 DZ) ; число смещений по горизонтали
      BTG (POLAR BT PI L) ; начальная точка оси графика
      J 0
      EG1 (POLAR BT 0 (* 4 L))
      BTG (POLAR BTG (* 1.5 PI) L)
      BTGR BTG
      MDP (/ (* P L) PI) ; приведенный момент движущих сил
      E 10 ; кинетическая энергия
      PIA (* PI 0.5)
      PIC (* PI 1.5)
      LK (* 2.5 L)
      LG (* 2 L)
```

Последние четыре строки определяют величину угла перемещения вверх (PIA), вниз (PIC), длину ведомого звена — LK и смещение базовой линии вниз — LG. Рассчитанные один раз, они экономят много машинного времени и упрощают написание программы;

- представление названий и создание графического изображения базовых линий для искомых параметров:

```
(REPEAT 4
  (IF (= I 3) (SETQ BTG BG1))
  (SETQ BTG (POLAR BTG PIC LG)
    BTT (POLAR BTG PIC (* 0.2 LG)))
  (COND
    ((= I 1) (SETQ TT "Кинетическая энергия агрегата"))
    ((= I 2) (SETQ TT "Приведенный момент на ведущем звене"))
    ((= I 3) (SETQ TT "Приведенный момент инерции J1P"))
    ((= I 4) (SETQ TT "Угловая скорость ведущего звена W")))
  )
  (COMMAND "TEXT" BTT "0" TT)
  (COMMAND "PLINE" BTG "W" 0.5 0.5 (POLAR BTG 0 (* 4 L)) "")
  (SETQ I (+ I 1))) ;
```

- создание графического изображения агрегата, расчет и представление искомых параметров. Поскольку изображение агрегата и все искомые параметры зависят от угла поворота ведущего звена, эту часть программы необходимо выполнять в цикле.

```
(REPEAT (+ 1 NZ)
  (SETQ Z (+ Z DZ)
    J (+ J 1)
    FI (+ FI DFI)
    OB (* (COS FI) L)
    AB (* (SIN FI) L)
    TB (POLAR BT 0 OB)
    TA (POLAR TB PIA AB)
    T3 (POLAR TB 0 LG)
    T4 (POLAR T3 0 (* 0.5 L))
    T31 (POLAR T3 PIA (* L 0.4))
    T32 (POLAR T3 PIC (* L 0.4))
    T41 (POLAR T4 PIA (* L 0.4))
    T42 (POLAR T4 PIC (* L 0.4)))
  (COMMAND "PLINE" BT "W" (* L 0.05) "" TA T3 T31 T41 T42
    T32 T3 "")
  (SETQ L1 (ENTLAST) I 1 BTG BTGR)
  (COMMAND "CIRCLE" TA (* L 0.1))
```

```

(SETQ L2 (ENTLAST))
(REPEAT 4
  (IF (= I 3) (SETQ BTG BG1))
  (IF (= R 1)
    (COND ((= I 2) (SETQ S MDP))
          ((= I 1) (SETQ S (* MDP FI)))
          ((= I 3) (SETQ S (+ J1 (/ (* G L L (SIN FI)
                                     (SIN FI)) GU)) J1P S))
          ((= I 4) (SETQ S (SQRT (/ (* 2 P L FI) (* J1P PI))))))
  )
  (SETQ BTG (POLAR BTG PIC LG))
  (IF (/= I 4) (SETQ S (/ S 100)) (SETQ S (* S 30)))
  (SETQ BT1 (POLAR BTG 0 Z)
    BT2 (POLAR BT1 PIA S)
    I (+ I 1))
  (COMMAND "PLINE" BT1 "W" 0.5 0.5 BT2 "")
  (IF (AND (= J (+ 1 NZ)) (/= I 5)) (COMMAND "TEXT" BT2 ""
    (RTOS (* S 100) 2 2)))
  (IF (AND (= J (+ 1 NZ)) (= I 5)) (COMMAND "TEXT" BT2 ""
    (RTOS (/ S 30) 2 2)))
  ) (ENTDEL L3) (ENTDEL L2) (ENTDEL L1)
  ) (PRIN1) (ENTDEL L3) (ENTDEL L2) (ENTDEL L1)
  )

```

Для того чтобы числовые значения искомых параметров представить в виде графика для любого дискретного положения ведущего звена, необходимо использовать счетчик J и ввести начальное значение J в первую функцию (SETQ...) и группу функций после последней функции (COMMAND...), например:

```
(IF (= J 1) (COMMAND "TEXT" BT2 "" (RTOS (* S 100) 2 2)))
```

6.1.4. Функция расчета и параметрического изображения агрегата и всех его характеристик

```
; *****
```

; (AGR L G P J1 N R) - функция создания параметрического изображения агрегата, расчета и изображения значений его динамических характеристик

; при различных режимах работы /разгон, установившийся режим работы, торможение/

; А р г у м е н т ы ф у н к ц и и:

; L - длина ведущего звена агрегата, мм

; G - масса ведомого звена агрегата, т

; P - усилие, воздействующее на ведомое звено, Н

; J1- момент инерции ведущего звена, кгмм кв.


```

; N - число оборотов ведущего звена в заданном режиме
; R - режим: 1 - разгон; 2 - установившийся режим; 3 - торможение
; Выводимые графические изображения:
; приведенный момент на ведущем звене
; кинетическая энергия агрегата E
; приведенный момент инерции J1P
; угловая скорость ведущего звена W
; *****
(DEFUN AGR (L G P J1 N R)
  (SETQ BT (LIST (* 2 L) (* 7 L))
    MXY (LIST (* 11 L) (* 8 L)))
  (COMMAND "LIMITS" "0,0" MXY "ZOOM" "A")
  (SETVAR "CMDECHO" 0) (SETVAR "BLIPMODE" 0)
  (COMMAND "STYLE" "" "" (* 0.15 L) "1" "" "" "" "")
  (COMMAND "CIRCLE" BT L
    "CIRCLE" BT (* L 0.1))
  (SETQ L3 (ENTLAST)) ; запоминание последнего построенного примитива
  (SETQ GU 9.81 I 1
    DFI (/ PI 6) FI (- 0 DFI)
    NZ (/ (* N 2 PI) DFI) NZ (+ 1 (FIX NZ))
    DZ (/ (* 4 L) NZ) Z (- 0 DZ)
    BTG (POLAR BT PI L) J 0
    BG1 (POLAR BT 0 (* 4.5 L)) PIC (* PI 1.5)
    BTG (POLAR BTG PIC L) LG (* 2 L)
    BTGR BTG PIA (* PI 0.5)
    LK (* 2.5 L) MDP (/ (* P L) PI)
    BG2 (POLAR BG1 PIA (* 0.8 L)) E 10)
  (SETQ T3 (POLAR BT 0 (* 1.7 L)) T4 (POLAR BT 0 (* 4.1 L))
    T31 (POLAR T3 PIA (* 0.5 L)) T32 (POLAR T3 PIC (* 0.5 L))
    T41 (POLAR T4 PIA (* 0.5 L)) T42 (POLAR T4 PIC (* 0.5 L)))
  (COMMAND "PLINE" T31 "W" 1 1 T41 T42 T32 "")
  (COND ((= R 1) (SETQ TTT "Режим разгона агрегата"))
    ((= R 2) (SETQ TTT "Установившийся режим работы"))
    ((= R 3) (SETQ TTT "Режим торможения агрегата")))
  (SETQ TTL (STRCAT "Длина кривошипа L=" (RTOS L 2 1) " MM")
    TTG (STRCAT "Масса поршня G=" (RTOS G 2 1) " Н")
    TTP (STRCAT "Усилие на поршне P=" (RTOS P 2 1) " Н")
    TTJ (STRCAT "Момент инерции поршня J1=" (RTOS J1 2 1)
      " КГММ КВ.")
    TTN (STRCAT "Число оборотов кривошипа, N=" (RTOS N 2 1)))
  (COMMAND "TEXT" BG2 "0" TTT
    "TEXT" (POLAR BG2 PIC (* 0.4 L)) "0" TTL
    "TEXT" (POLAR BG2 PIC (* 0.7 L)) "0" TTG
    "TEXT" (POLAR BG2 PIC (* 1.0 L)) "0" TTP

```

```

      *TEXT* (POLAR BG2 PIC (* 1.3 L)) "0" TTJ
      *TEXT* (POLAR BG2 PIC (* 1.7 L)) "0" TTN )
(REPEAT 4
  (IF (= I 3) (SETQ BTG (POLAR BG1 PIC L)))
  (SETQ BTG (POLAR BTG PIC LG)
    BTT (POLAR BTG PIC (* 0.8 L)))
(COND
  ((= I 1) (SETQ TT "Кинетическая энергия агрегата"))
  ((= I 2) (SETQ TT "Приведенный момент на ведущем звене"))
  ((= I 3) (SETQ TT "Приведенный момент инерции J1P"))
  ((= I 4) (SETQ TT "Угловая скорость ведущего звена W")))
(COMMAND "TEXT" BTT "0" TT)
(COMMAND "PLINE" BTG "W" 0.5 0.5 (POLAR BTG 0 (* 4 L)) "")
(SETQ I (+ I 1))
(REPEAT (+ 1 NZ)
  (SETQ Z (+ Z DZ)
    J (+ J 1)
    FI (+ FI DFI)
    OB (* (COS FI) L)
    AB (* (SIN FI) L)
    TB (POLAR BT 0 OB)
    TA (POLAR TB PIA AB)
    T3 (POLAR TB 0 LK)
    T4 (POLAR T3 0 (* 0.5 L))
    T31 (POLAR T3 PIA (* L 0.4))
    T32 (POLAR T3 PIC (* L 0.4))
    T41 (POLAR T4 PIA (* L 0.4))
    T42 (POLAR T4 PIC (* L 0.4)))
(COMMAND "PLINE" BT "W" (* L 0.05) "" TA T3 T31 T41 T42
  T32 T3 "")
(SETQ L1 (ENTLAST) I 1 BTG BTGR)
(COMMAND "CIRCLE" TA (* L 0.1))
(SETQ L2 (ENTLAST))
(REPEAT 4
  (IF (= I 3) (SETQ BTG (POLAR BG1 PIC L)))
  (IF (= R 1)
    (COND
      ((= I 2) (SETQ S MDP))
      ((= I 1) (SETQ S (* MDP FI)))
      ((= I 3) (SETQ S (+ J1 (/ (* G L L (SIN FI) (SIN FI)) GU)) J1P S))
      ((= I 4) (SETQ S (SQRT (/ (* 2 P L FI) (* J1P PI))))))
  (IF (= R 2)
    (PROGN
      (IF (> FI (* 2 PI)) (SETQ FI (- FI (* 2 PI))))

```

```

(COND
  ((= I 2) (IF (AND (>= FI 0) (<= FI PI))
    (SETQ MP (* (* P L) (- (/ 1 PI) (SIN FI))))
    (SETQ MP (/ (* P L) PI)))
    (SETQ S MP))
  ((= I 1) (IF (AND (>= FI 0) (<= FI PI))
    (SETQ E (* P L (+ (/ FI PI) (COS FI) -1 (* 2 N))))
    (SETQ E (* P L (+ (/ FI PI) -2 (* 2 N)))))
    (SETQ S E))
  ((= I 3) (SETQ J1P (+ J1 (/ (* G L L (SIN FI)
    (SIN FI)) GU))) (SETQ S J1P))
  ((= I 4) (SETQ W (SQRT (/ (* 2 E) J1P)) S W))))
(IF (AND (= R 3) (/= E 0))
  (COND
    ((= I 2) (IF (OR (<= FI PI) (AND (>= FI (* 2 PI)) (<= FI (* 3 PI))))
      (SETQ MCP (- 0 (* P L (SIN FI))) S MCP)
      (SETQ MCP 0 S MCP)))
    ((= I 1) (IF (<= FI PI)
      (SETQ E (* P L (+ (* 2 N) (COS FI) -1)) S E))
      (IF (AND (>= FI PI) (<= FI (* 2 PI)))
        (SETQ E (* 2 P L (- N 1)) S E))
        (IF (AND (>= FI (* 2 PI)) (<= FI (* 3 PI)))
          (SETQ E (* P L (+ (* 2 N) -3 (COS FI))) S E)))
    ((= I 3) (SETQ J1P (+ J1 (/ (* G L L (SIN FI)
      (SIN FI)) GU)) S J1P))
    ((= I 4) (SETQ W (SQRT (/ (* 2 E) J1P)) S W))))
  (SETQ BTG (POLAR BTG PIC LG))
  (IF (/= I 4) (SETQ S (/ S 100)) (SETQ S (* S 30)))
  (SETQ BT1 (POLAR BTG 0 Z)
    BT2 (POLAR BT1 PIA S)
    I (+ I 1))
  (COMMAND "PLINE" BT1 "W" 0.5 0.5 BT2 "")
  (IF (AND (= J (+ 1 NZ)) (/= I 5)) (COMMAND "TEXT" BT2 ""
    (RTOS (* S 100) 2 2)))
  (IF (AND (= J (+ 1 NZ)) (= I 5)) (COMMAND "TEXT" BT2 ""
    (RTOS (/ S 30) 2 2)))
) (ENTDEL L3) (ENTDEL L2) (ENTDEL L1)
) (PRIN1) (ENTDEL L3) (ENTDEL L2) (ENTDEL L1)
)

```

Функцию (AGR...) поместим в файл, например, AGR.LSP.

Для исследования режима разгона агрегата необходимо ввести в функцию соответствующие исходные данные, например:

(AGR 100.0 10.0 50.0 5000.0 2.0 1) ; расчет режима разгона

Результат выполнения функции (AGR...) в режиме разгона агрегата представлен на рис. 6.2.

Для исследования установившегося режима работы агрегата необходимо ввести соответствующие исходные данные в функцию, например:

(AGR 100.0 10.0 50.0 5000.0 2.0 2) ; расчет установившегося режима

Результат выполнения функции (AGR...) в установившемся режиме представлен на рис. 6.3.

Для исследования режима торможения агрегата необходимо ввести соответствующие исходные данные в функцию, например:

(AGR 100.0 10.0 50.0 5000.0 2.0 3) ; расчет режима торможения

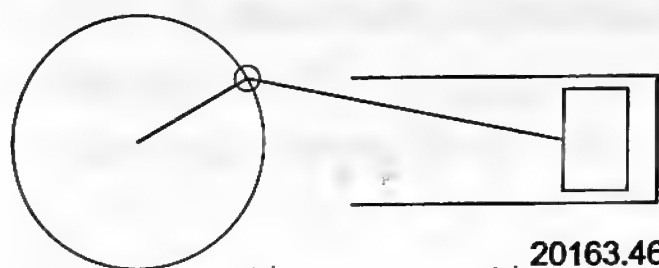
Результат выполнения функции (AGR...) в режиме торможения представлен на рис. 6.4.

6.2. Расчет и создание параметрического изображения нагрузок в фермах

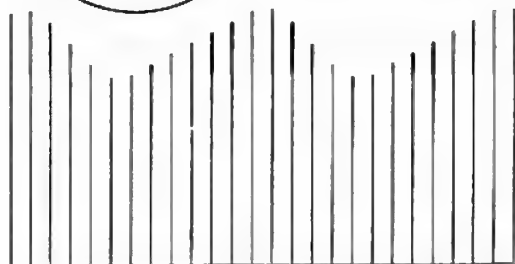
Использование системы AutoCAD и языка AutoLISP для расчета и создания параметрического изображения нагрузок, а точнее, расчета и построения линий влияния для плоских стержневых конструкций, рассмотрим на примере стрелы башенного крана (подвесного моста, консоли и т. д.).



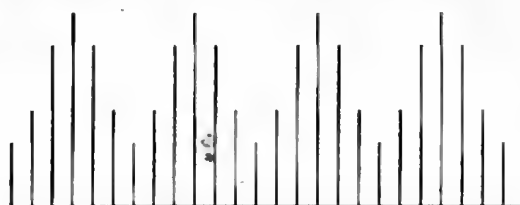
Рис. 6.2. Режим разгона агрегата



Установившийся режим работы
 Длина кривошипа $L=100$ мм
 Масса поршня $G=10$ Н
 Усилие на поршне $P=50$ Н
 Момент инерции поршня $J_1=5000$ кгмм кв.
 Число оборотов кривошипа, $N=2$



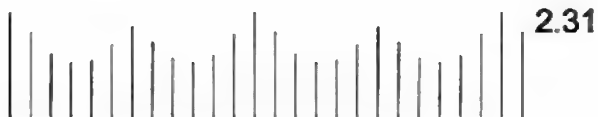
Кинетическая энергия агрегата



Приведенный момент инерции J_1P

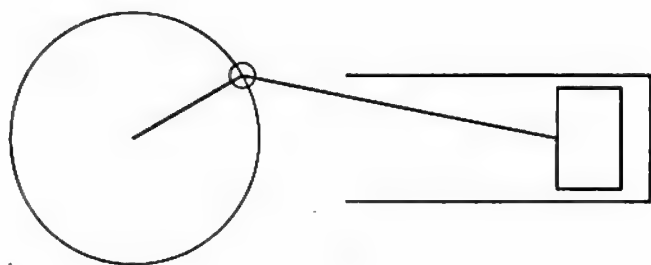


Приведенный момент на ведущем звене

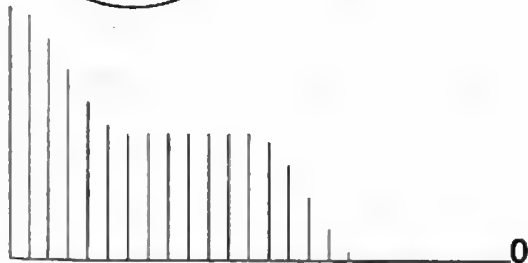


Угловая скорость ведущего звена W

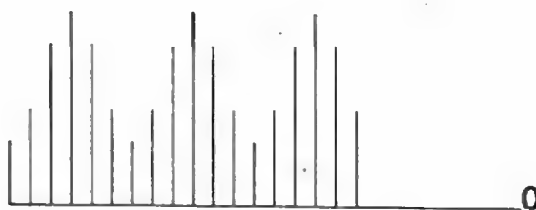
Рис. 6.3. Установившийся режим работы агрегата



Режим торможения агрегата
 Длина кривошипа $L=100$ мм
 Масса поршня $G=10$ Н
 Усилие на поршне $P=50$ Н
 Момент инерции поршня $J_1=5000$ кгмм кв.
 Число оборотов кривошипа, $N=2$



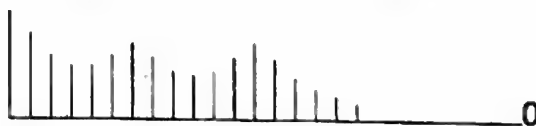
Кинетическая энергия агрегата



Приведенный момент инерции J_1P



Приведенный момент на ведущем звене



Угловая скорость ведущего звена W

Рис. 6.4. Режим торможения агрегата

6.2.1. Постановка задачи

Разработать алгоритм и программу (функцию) автоматизации расчета и создания параметрического изображения линий влияния для стержней заданной секции стрелы, представленной на рис. 6.5.

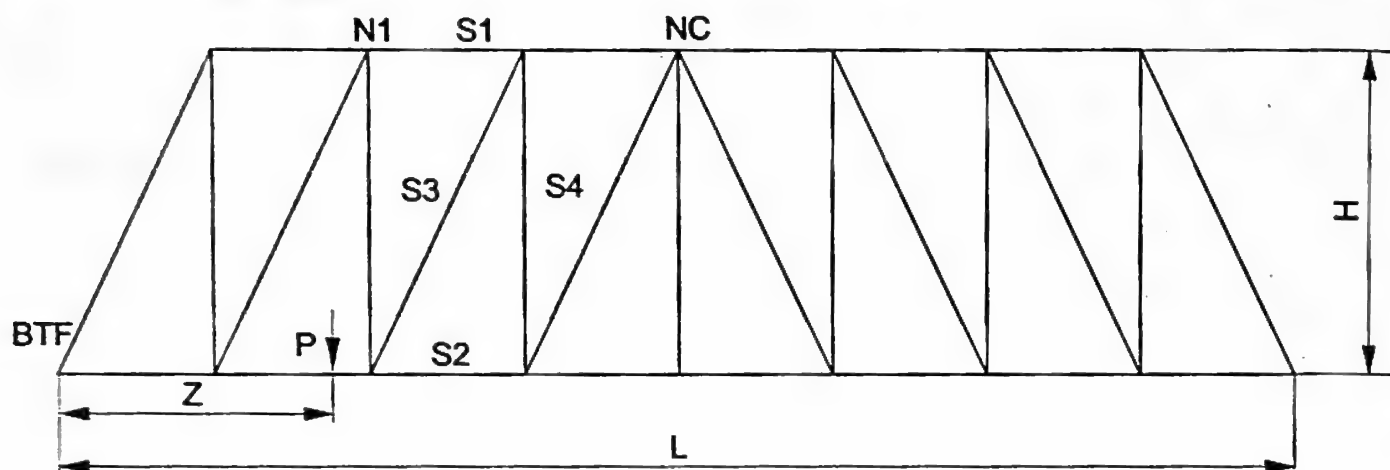


Рис. 6.5. Вид фронтальной проекции стрелы

В качестве языка программирования использовать AutoLISP. Основными исходными данными для решения задачи являются: базовая точка крепления стрелы – BTF; длина стрелы – L ; высота стрелы – H ; число секций в стреле – N ; номер секции подвеса стрелы – NC ; номер узла, который предшествует расчетной секции, – $N1$; число дискретных перемещений единичной силы вдоль нижнего пояса – NZ ; схема подвески стрелы (см. рис. 6.6).

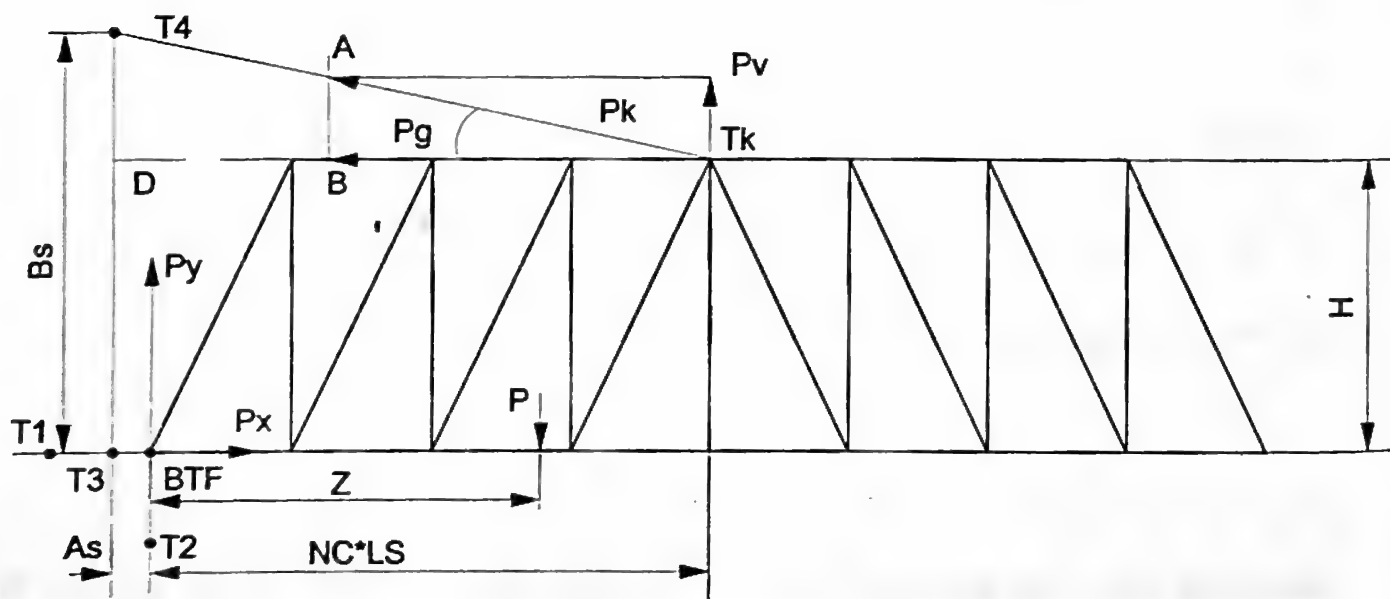


Рис. 6.6. Схема подвески стрелы

6.2.2. Выявление основных особенностей, взаимосвязей и количественных закономерностей

Исходя из конструкции стрелы (рис. 6.5), в расчетной секции можно выделить следующие виды стержней: стержень верхнего пояса стрелы, обозначим его S_1 ; стержень нижнего пояса стрелы – S_2 ; раскос – S_3 ; стойку – S_4 .

Усилия в стержнях стрелы зависят от схемы подвески стрелы, которая представлена на рис. 6.6. Определяющими параметрами здесь являются координаты точек Т4 и ТК.

Для определения местоположения точки Т4 достаточно знать координаты базовой точки ВТФ и относительные смещения AS и BS искомой точки Т4 относительно базовой точки ВТФ. Местоположение необходимых точек для проведения расчета и создания параметрического изображения линий влияния определим с помощью полярной системы координат.

Чтобы провести необходимые расчеты, необходимо определить значения реакций в опоре ВТФ и усилия, действующие на стрелу в точке крепления каната ТК.

Для расчета последних возьмем сумму моментов действующих сил относительно базовой точки:

$$P \cdot Z - P_G \cdot H - P_v \cdot NC \cdot LS = 0$$

Используя подобие треугольников ТК АВ и ТК D Т4, определим P_v / P_G :

$$\frac{P_v}{P_G} = \frac{BS - H}{AS + NC \cdot LS} = A; \quad P_v = P_G \frac{BS - H}{AS + NC \cdot LS} = P_G \cdot A$$

$$P \cdot Z - P_G \cdot H - P_G \frac{BS - H}{AS + NC \cdot LS} NC \cdot LS = 0; \quad P_G = \frac{P \cdot Z}{H + A \cdot NC \cdot LS}$$

6.2.3. Разработка последовательности действий для расчета и параметрического изображения линий влияния

Расчет и создание параметрического изображения линий влияния включает следующие основные этапы:

- *определение местоположения точек крепления стрелы, каната и графическое изображение их на чертеже.* Местоположение точек Т1, Т2, Т4, ТК можно найти, используя полярную систему координат. Местоположение точек Т1 и Т2 можно определить путем перемещения базовой точки ВТФ под углом ρ_1 и $1,5 \rho_1$ соответственно на расстояние $0,3 H$. Местоположение точки Т4 определяется при перемещении от базовой точки ВТФ под углом ρ_1 на расстояние AS, а затем от этой же точки под углом $\rho_1/2$ на расстояние BS. Местоположение точки ТК можно

установить путем перемещения точки ВТФ под углом 0 на расстояние $NC \cdot LS$, а затем под углом $\pi/2$ на расстояние H . Для создания графического изображения каната следует соединить точки Т4 и ТК;

- *определение точек расчетной секции* (рис. 6.7). Местоположение точки Т1 расчетной секции определяется путем перемещения базовой точки на расстояние $N1 \cdot LS$ под углом 0 . Местоположение остальных точек (Т2, Т3, Т4) можно определить по рис. 6.7, если перемещаться в направлениях, указанных стрелками;
- *подготовка к расчету*. Введем переменную I , определяющую стержень сечения, для которого строится линия влияния:
если $I = 1$, стержень верхнего пояса – S_1 ;
если $I = 2$, стержень нижнего пояса – S_2 ;
если $I = 3$, раскос – S_3 ;
если $I = 4$, стойка – S_4 .

Определим шаг перемещения единичной силы $DZ = L / NZ$, исходя из числа перемещений единичной силы – NZ ;

- *выделение на чертеже расчетного стержня в процессе построения линии влияния и определение основных составляющих действующих сил*. Построим отрезок – основание линии влияния, на котором будет строиться линия влияния. Толстой линией выделим стержень в расчетной секции, для которого проводится расчет. Изобразим единичную силу в виде стрелки (она дискретно перемещается по нижнему поясу стрелы).

Определим реакции в точке опоры стрелы и усилия в точке ее подвеса:

$$P_Y = P - P_v; P_v = P_G \cdot A; P_x = P_G; P = 1$$

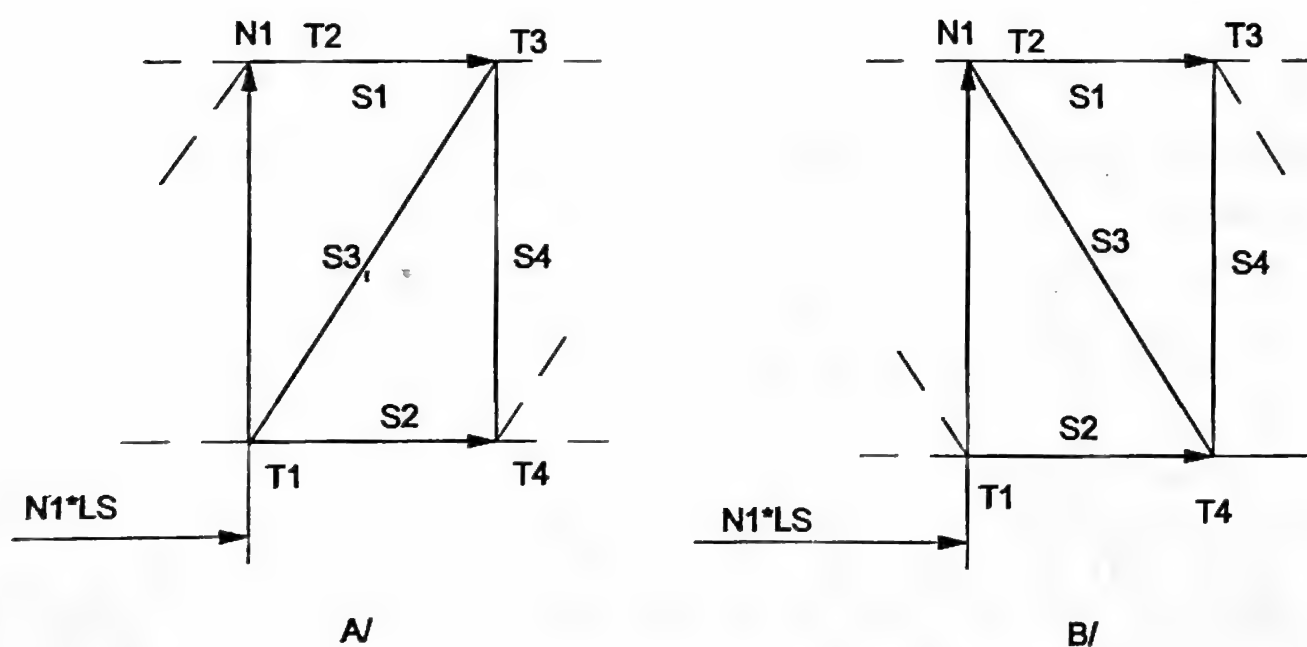


Рис. 6.7. Схемы местоположения точек расчетной секции: а) - расположенной до точки крепления каната; б) - расположенной после точки крепления каната

$$P_G = \frac{P \cdot Z}{H + A \cdot NC \cdot LS} = \frac{Z}{H + A \cdot NC \cdot LS}$$

- *расчет усилий в стержнях секции.* Для расчета усилий в стержнях стрелы используем метод Риттера, согласно которому ферма рассекается на две части таким образом, чтобы в сечении было не более трех стержней с неизвестными усилиями. Отбрасываем отсеченную часть фермы и оставшуюся ее часть рассматриваем в равновесии под действием приложенных к ней внешних сил и усилий, заменяющих действие расчлененных стержней. Для этой части фермы получим три уравнения равновесия, в которые войдут три искоемых усилия.

Начнем с определения усилия в стержне верхнего пояса S_1 . Возьмем сумму моментов всех сил относительно узла C , в котором пересекаются два других стержня (рис. 6.8).

Проведем сечение, пересекающее три стержня, и рассмотрим равновесие левой части фермы под действием сил: реакций опоры P_x и P_y в узле O , единичной силы P (для схемы, представленной на рис. 6.8, а) и действующей на расстоянии Z от опоры O . При этом все усилия в стержне направим от узлов, то есть таким образом, как если бы они были растянуты. При этом возможны два случая: расчетная секция расположена до точки подвеса стрелы и после этой точки (рис. 6.6).

Каждый из этих случаев, в зависимости от места расположения единичной силы, имеет два варианта (рис. 6.8, а, б).

Единичная сила действует перед узлом C . $Z < N1 \cdot LS$. В таком случае сумма моментов относительно точки C составит:

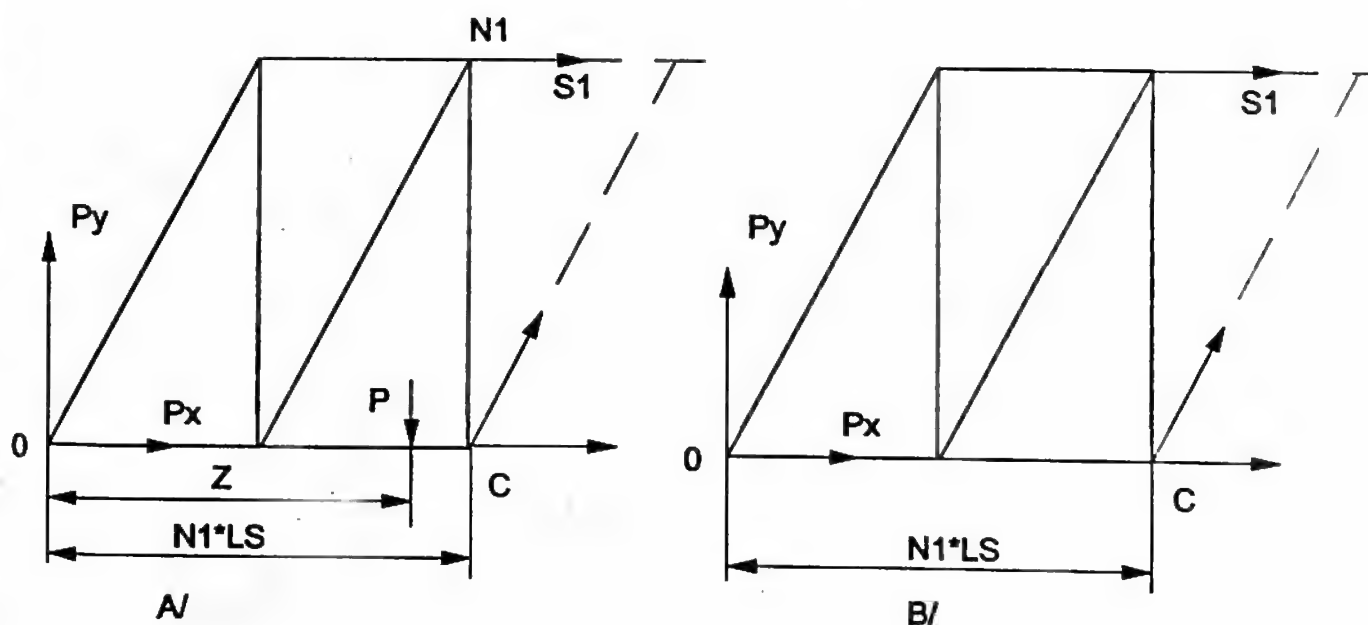


Рис. 6.8. Схемы расчета усилия в стержне S_1 для секций, расположенных до точки подвеса стрелы

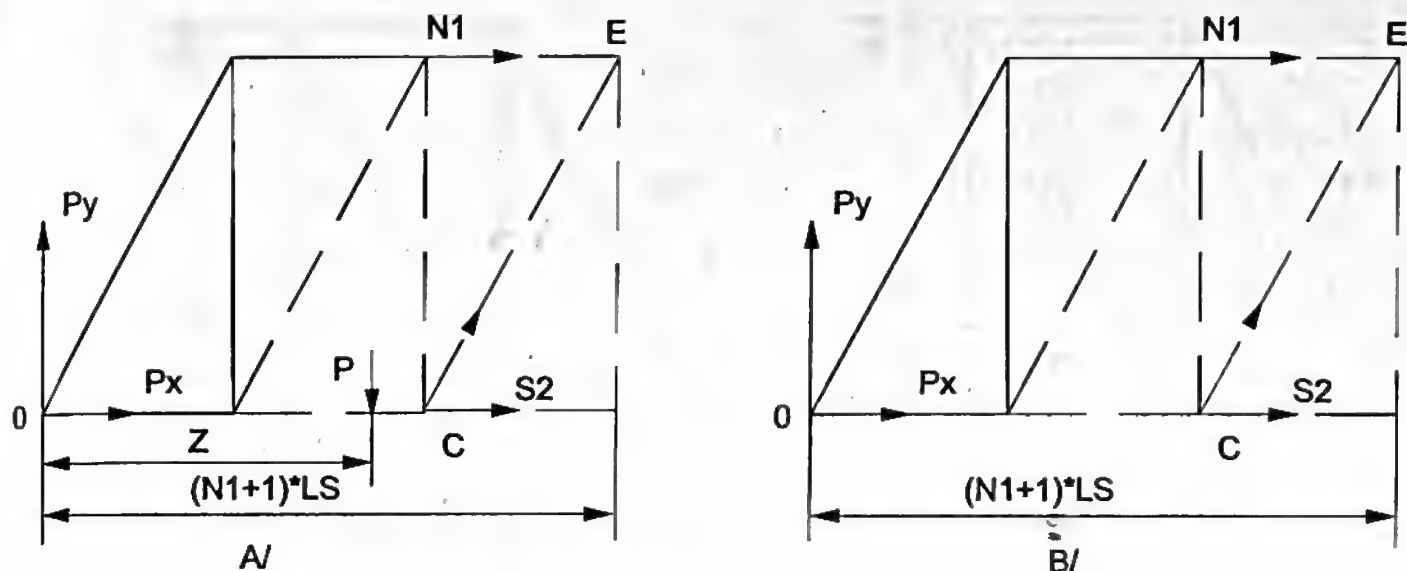


Рис. 6.10. Схемы расчета усилия в стержне S_2 для секций, расположенных до точки подвеса

$$S_2 * H - P_y * (N1 + 1) * LS = 0, S_2 = P_y * (N1 + 1) * LS / H$$

В том случае, когда расчетная секция расположена после точки подвеса стрелы (рис. 6.11), возможны два варианта:

а) единичная сила находится до узла $N1$ и сумма моментов относительно этого узла составит:

$$\sum M = 0, S_2 * H = 0,$$

следовательно, $S_2 = 0$, т. к. $H \neq 0$

б) единичная сила находится после узла $N1$ и сумма моментов составит:

$$\sum M = 0, S_2 * H + P * (Z - N1 * LS) = 0,$$

следовательно, $S_2 = -P * (Z - N1 * LS) / H$

Рассчитаем усилия в раскосе S_3 секций, расположенных до точки подвеса стрелы (рис. 6.12).

Если два из трех стержней сечения параллельны, то одна из точек Риттера удаляется в бесконечность. Для определения усилия в непараллельном

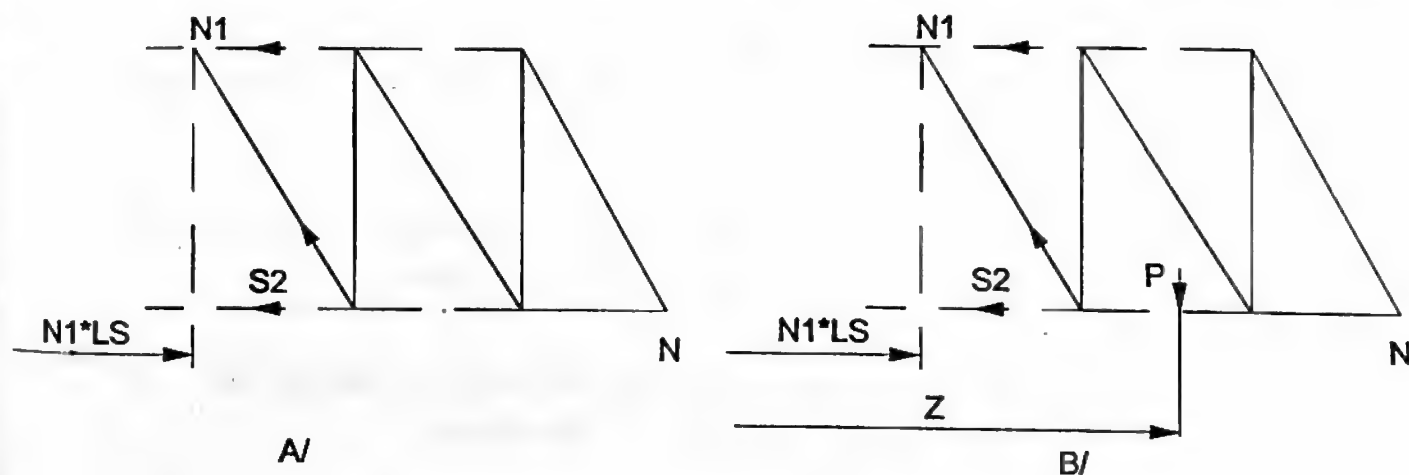


Рис. 6.11. Схемы расчета усилия в стержне S_2 для секций, расположенных после точки подвеса

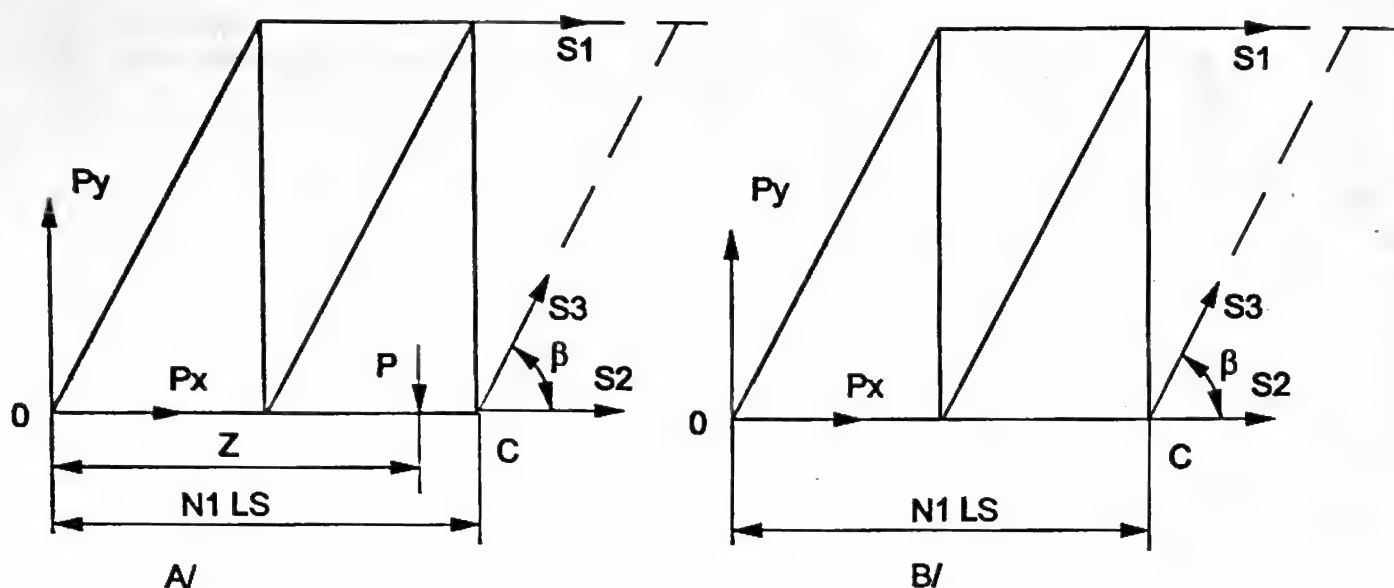


Рис. 6.12. Схемы расчета усилий в раскосе S_3 для секций, расположенных перед точкой подвеса стрелы

стержне вместо уравнения моментов можно взять сумму проекций всех сил в направлении, перпендикулярном параллельным стержням. В нашей задаче стержни S_1 и S_2 параллельны. При этом возможны два варианта:

- а) единичная сила находится до рассматриваемого раскоса $Z < N1 \cdot LS$, тогда сумма проекций всех сил на ось Y составит:

$$P_Y - P + S_3 \sin \beta = 0; \sin \beta = \frac{H}{\sqrt{H^2 + LS^2}},$$

$$\text{откуда } S_3 = \frac{(P - P_Y) \sqrt{H^2 + LS^2}}{H}$$

- б) единичная сила находится после рассматриваемого раскоса, тогда сумма проекций всех сил на ось Y составит:

$$P_Y + S_3 \sin \beta = 0; S_3 = -P_Y \frac{\sqrt{H^2 + LS^2}}{H}$$

Рассчитаем усилия в стержне S_3 для секций, расположенных после точки подвеса стрелы (рис. 6.13).

Возможны два варианта:

- а) единичная сила находится до рассматриваемого раскоса ($Z \leq N1 \cdot LS$), тогда сумма проекций всех сил на ось Y составит:

$$S_3 \sin \beta = 0, S_3 = 0$$

- б) единичная сила находится после рассматриваемого раскоса ($Z > N1 \cdot LS$), тогда сумма проекций всех сил на ось Y составит:

$$S_3 \sin \beta = P, S_3 = P \frac{\sqrt{H^2 + LS^2}}{H}$$

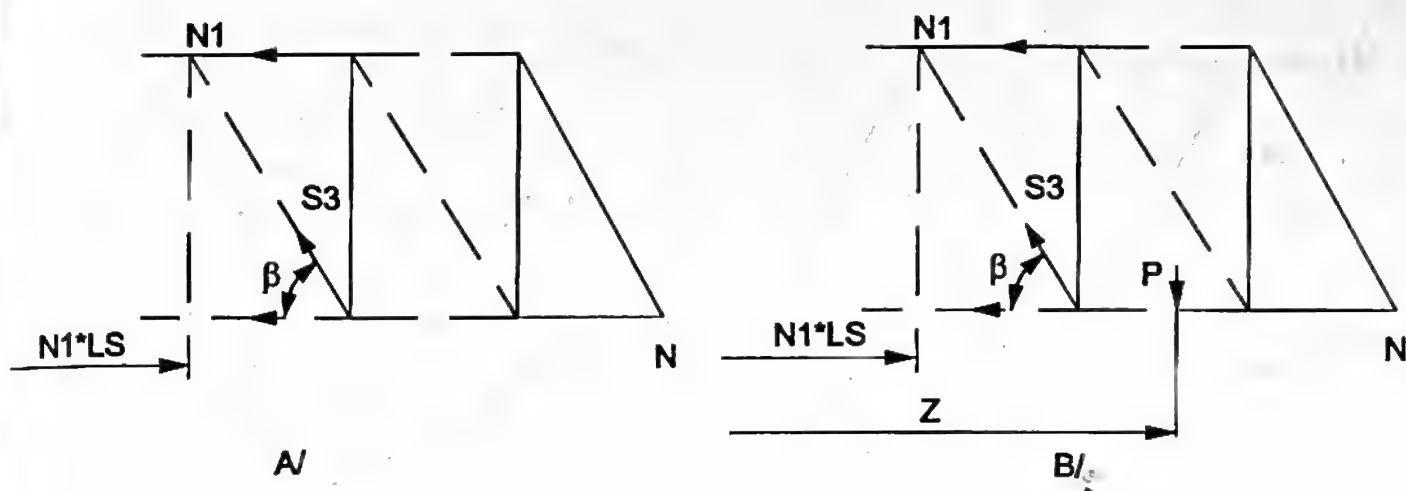


Рис. 6.13. Схемы расчета усилия в раскосе S_3 для секций, расположенных после точки подвеса стрелы

Рассчитаем усилия в стойке S_4 для секций, расположенных до точки подвеса стрелы (рис. 6.14).

Возможны два варианта:

- а) единичная сила находится до узла $(N1 + 1)$, то есть $Z \leq (N1 + 1) * LS$.

В таком случае сумма проекций всех сил составляет:

$$P_y - P - S_4 = 0,$$

$$\text{откуда } S_4 = P_y - P = P_y - 1$$

- б) единичная сила находится после точки $(N1 + 1)$, то есть $Z > (N1 + 1) * LS$. В этом случае сумма проекций всех сил составит: $P_y - S_4 = 0$,

$$\text{откуда } S_4 = P_y$$

Рассчитаем усилия в стойке S_4 секций, расположенных после точки подвеса стрелы (рис. 6.15).

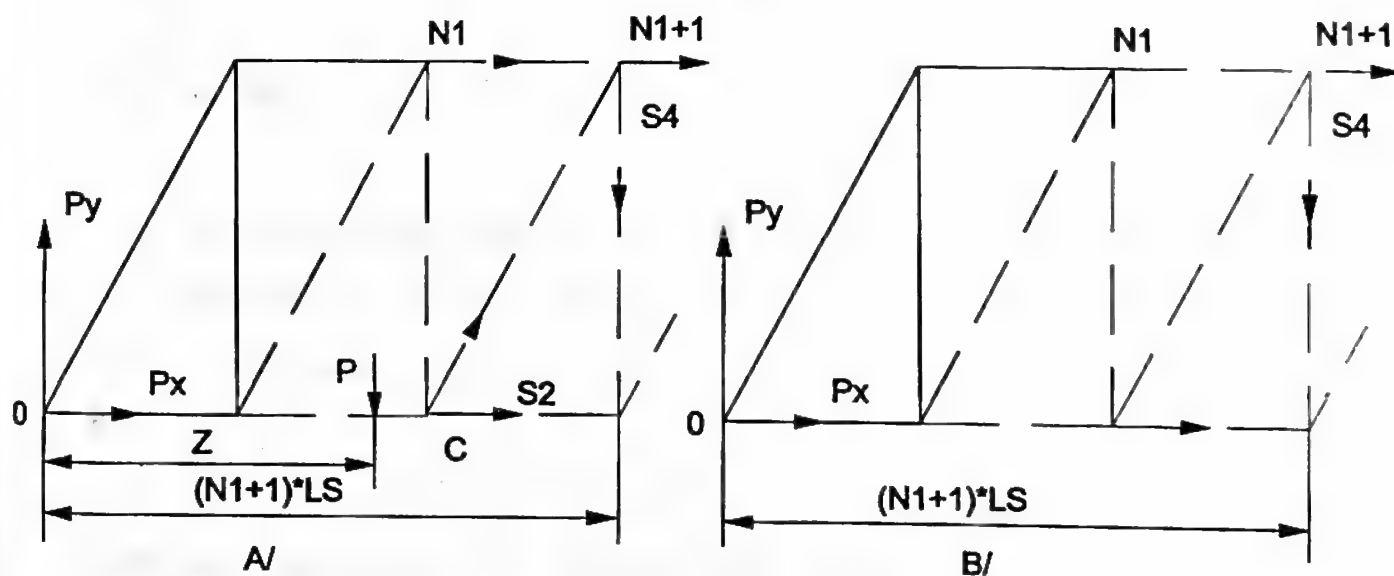


Рис. 6.14. Схемы расчета усилия в стойке S_4 для секций, расположенных до точки подвеса стрелы

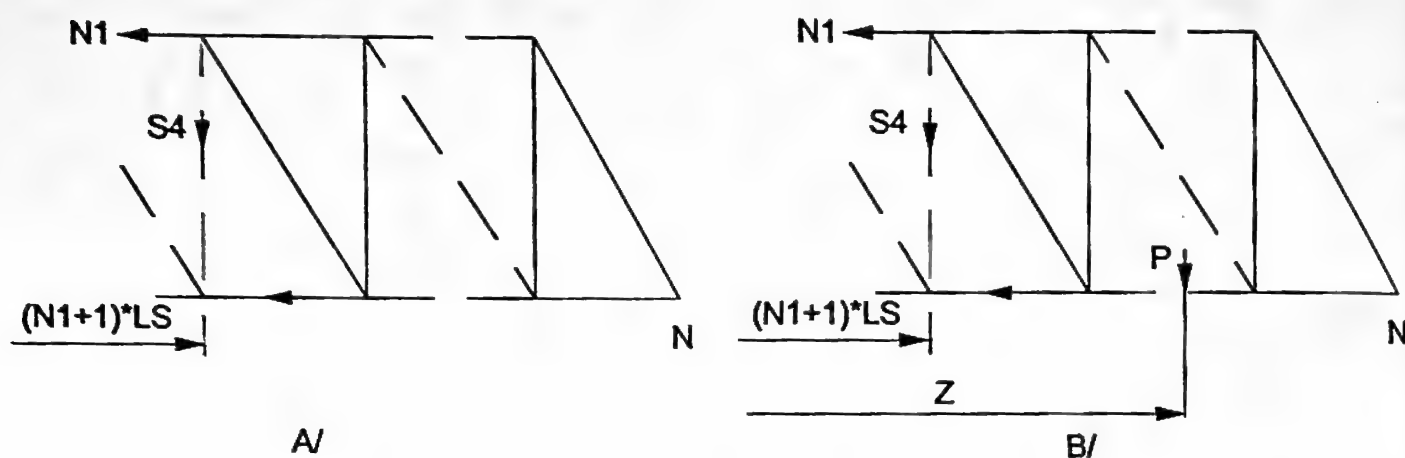


Рис. 6.15. Схемы определения усилия в стойке S_4 в сечениях, расположенных после точки подвеса стрелы

При этом возможны два варианта:

- единичная сила находится до рассматриваемой стойки ($Z < (N1 + 1) \cdot LS$) и сумма проекций всех сил составляет на ось Y: $S_4 = 0$;
 - единичная сила находится после рассматриваемой стойки и тогда при сумме проекций всех сил на ось Y: $S_4 = P$;
- создание графического изображения усилия (линии влияния) в рассматриваемом стержне. Перпендикулярно отрезку – основанию линии влияния – откладываем усилие соответствующего стержня с учетом положения единичной силы на стреле. При этом знак усилия в стержне учитывается автоматически. Масштаб усилий в стержнях пропорционален 0.3 Н. Одновременно производится удаление графического изображения единичной силы на стреле до перехода ее в новую позицию, а также временное удаление стержня, для которого осуществляется расчет. Это необходимо, чтобы увидеть, для какого стержня строится линия влияния. При перемещении единичной силы на новое место на стреле графическое изображение расчетного стержня восстанавливается.

6.2.4. Разработка комплекса функций для расчета и параметрического изображения линий влияния

Разработка комплекса функций AutoLISP включает следующие основные этапы:

- определение местоположения точек крепления стрелы, каната и создание на чертеже их графического изображения. Для разработки этой части программы используем функции (POLAR...), а также команды "PLINE" (построение линий) и "CIRCLE" (построение окружностей). Соответствующая часть программы будет выглядеть следующим образом:

```
(SETQ T1 (POLAR BTF PI (* 0.3 H))
      T2 (POLAR BTF (* PI 1.5) (* 0.3 H))
      T3 (POLAR BTF PI AS)
      T4 (POLAR T3 (* PI 0.5) BS)
      TK1 (POLAR BTF 0 (* NC LS))
      TK (POLAR TK1 (* PI 0.5) H)
```

```
)
(COMMAND "PLINE" T1 "W" 1 1 BTF T2 ""
  "PLINE" T4 "W" 2 2 TK ""
  "CIRCLE" T1 3 "CIRCLE" T2 3
  "CIRCLE" T4 3 "CIRCLE" TK 3)
```

• *определение точек расчетной секции:*

```
(SETQ T1 (POLAR BTF 0 (* N1 LS))
      T2 (POLAR T1 (* PI 0.5) H)
      T3 (POLAR T2 0 LS)
      T4 (POLAR T1 0 LS)
```

• *подготовка к расчету:*

```
(SETQ A (/ (- BS H) (+ AS (* NC LS)))
      N1LS (* N1 LS)
      N11LS (* (+ N1 1) LS)
      I 1
      BT BTF
      DZ (/ (FLOAT L) NZ)
      SINB (/ H (SQRT (+ (* H H) (* LS LS))) )
```

• *выделение на чертеже в процессе построения линии влияния расчетного стержня и определение основных составляющих действующих сил.*

С помощью функции (**REPEAT 4...**) сформируем цикл для расчета стержней S_1, S_2, S_3, S_4 и построим изображение основания линии влияния:

```
(REPEAT 4
  (SETQ BT (POLAR BT (* PI 1.5) (* 0.7 H))
        BTK (POLAR BT 0 L)
        Z 0)
  (COMMAND "PLINE" BT "W" 1 1 BTK "")
  (SETQ BTT (POLAR BT (* PI 0.5) (* 0.3 H)))
```

Для выделения нужного стержня и перемещения единичной силы в этом же цикле сформируем цикл и запомним последние построенные примитивы с помощью функции (**ENTLAST...**):

```
(REPEAT NZ
  (COND ((= I 1) (COMMAND "PLINE" T2 "W" 2 2 T3 "")))
```



```

( (= I 2) (COMMAND "PLINE" T1 "W" 2 2 T4 ""))
( (= I 3) (IF (< N1 NC)
  (COMMAND "PLINE" T1 "W" 2 2 T3 "")
  (COMMAND "PLINE" T2 "W" 2 2 T4 "")))
( (= I 4) (COMMAND "PLINE" T3 "W" 2 2 T4 ""))
)
(SETQ Z (+ DZ Z))
L2 (ENTLAST)
BTP (POLAR BTP 0 Z)
PG (/ Z (+ (* A NC LS) H))
P 1
PV (* PG A)
PY (- P PV)
PX PG
P1 (/ (* P (- N1LS Z)) LS)
P2 (/ (* P (- Z N1LS)))
)
(COMMAND "PLINE" BTP "W" 0.5 5
(POLAR BTP (* PI 0.5) (* 0.3 H)) "")
(SETQ L1 (ENTLAST))

```

Все команды **"PLINE"**, кроме последней, предназначены для выделения того стержня секции, которому соответствует текущий номер цикла (**REPEAT 4...**). В переменной L2 запоминается последний выделенный примитив, то есть стержень. На нижнем поясе стрелы последняя команда **"PLINE"** показывает единичную силу в виде стрелки, а переменная L1 с помощью функции (**ENTLAST...**) запоминает изображенный примитив.

- *расчет усилий в стержнях секций.* Выделение того или иного стержня для расчета произведем с помощью функции (**COND...**).

```

(COND ((= I 1)
  (IF (< N1 NC)
    (IF (<= Z N1LS)
      (SETQ S (/ (- (- N1LS Z) (* PY N1LS)) H))
      (SETQ S (/ (- 0 (* PY N1LS)) H))
    )
    (IF (<= Z N11LS)
      (SETQ S 0)
      (SETQ S (/ (- Z N11LS) H))))
  )
  ((= I 2)
    (IF (< N1 NC)
      (IF (<= Z N11LS)
        (SETQ S (/ (- (* PY N11LS) (- N11LS Z)) H))

```

```

    (SETQ S (/ (* PY N1LS) H))
  )
  (IF (<= Z N1LS)
    (SETQ S 0)
    (SETQ S (/ (- N1LS Z) H))))
  )
  ((= I 3)
  (IF (< N1 NC)
    (IF (<= Z N1LS)
      (SETQ S (/ (- 1 PY) SINB))
      (IF (AND (> Z N1LS) (< Z N11LS))
        (SETQ S (/ (- P1 PY) SINB))
        (SETQ S (/ (- 0 PY) SINB)))
    )
    (IF (<= Z N11LS)
      (SETQ S 0)
      (IF (AND (> Z N1LS) (< Z N11LS))
        (SETQ S (/ P1 SINB))
        (SETQ S (/ 1 SINB))))
    )
  )
  ((= I 4)
  (IF (< N1 NC)
    (IF (<= Z N11LS)
      (SETQ S (- PY 1))
      (SETQ S PY)
    )
    (IF (< Z N11LS)
      (SETQ S 0)
      (SETQ S 1))))
  )

```

• создание графического изображения усилия (линии влияния) в рассматриваемом стержне:

```

(SETQ BT1 (POLAR BT 0 Z)
  S (* S (* 0.3 H))
  BT2 (POLAR BT1 (* PI 0.5) S))
(COMMAND "PLINE" BT1 "W" 2 2 BT2 "")
  (ENTDEL L2)
  (ENTDEL L1)
)
(SETQ I (+ I 1))
(SETQ ZN (RTOS (/ S (* 0.3 H)) 2 2))
(COMMAND "TEXT" BT2 "0" ZN)
)

```

```
(IF (< N1 NC) (COMMAND "PLINE" T2 "W" 2 2 T3 T4 T1 T3 "")
  (COMMAND "PLINE" T1 "W" 2 2 T4 T3 T2 T4 ""))
(PRIN1)
)
```

В заключение осталось сформировать начало и конец функции, с помощью которой будет строиться линия влияния, например:

```
(DEFUN STRELA (XF YF L H N NC BS AS N1 NZ)
  (SETVAR "CMDECHO" 0)
  (SETVAR "BLIPMODE" 0)
  (COMMAND "LIMITS" "0,0" "400,300" "ZOOM" "A")
  (COMMAND "STYLE" "" "" "5" "1.5" "" "" "" "")
  (SFERMA1)
  . . . . .
)
```

Ниже представлен комплекс функций для создания параметрического изображения фронтальной проекции стрелы заданного вида и построения линий влияний для стержней заданной секции стрелы. Функция (SFERMA) строит фронтальную проекцию стрелы заданного вида, рассмотренную в главе 3.

6.2.5. Комплекс функций для расчета и параметрического изображения линий влияния стрелы

```
; *****
; Комплекс функций для создания параметрического изображения
; фронтальной проекции стрелы заданного вида и построения
; линий влияний для стержней заданной секции стрелы
; *****
; (SFERMA1 XF YF L H N) - изображение фронтальной проекции
; (STRELA XF YF L H N NC BS AS N1 NZ) - расчет и изображение
; линий влияния для стержней заданной секции стрелы
; А р г у м е н т ы  ф у н к ц и и:
; XF - базовая точка стрелы по оси X
; YF - базовая точка стрелы по оси Y
; L, H, N - длина, высота и число секций в стреле
; NC - номер секции подвеса стрелы
; AS, BS - точка крепления каната стрелы по оси X и Y
; N1 - номер секции, предшествующий расчетной
; NZ - число дискретных перемещений единичной силы по стреле
; *****
(DEFUN SFERMA1 ()
  (SETQ BTF (LIST XF YF)
    LS (/ L N)
```

```

    T2 (POLAR BTF 0 LS)
    T1 (POLAR T2 (/ PI 2) H)
    I 0
    NS (- N 2))
(COMMAND "PLINE" T2 "W" 0.5 0.5 BTF T1 "C")
(REPEAT NS
  (SETQ T3 (POLAR T2 0 LS)
    T4 (POLAR T1 0 LS)
    I (+ I 1))
  (IF (< I NC)
    (COMMAND "PLINE" T1 "W" 0.5 0.5 T4 T3 T2 T4 "")
    (COMMAND "PLINE" T2 "W" 0.5 0.5 T3 T4 T1 T3 ""))
  )
  (SETQ T2 T3
    T1 T4)
)
(SETQ TN (POLAR T2 0 LS))
(COMMAND "PLINE" T4 "W" 0.5 0.5 TN T3 "")
)
(DEFUN STRELA (XF YF L H N NC BS AS N1 NZ)
  (SETVAR "CMDECHO" 0)
  (SETVAR "BLIPMODE" 0)
  (COMMAND "LIMITS" "0,0" "400,300" "ZOOM" "ALL" "" )
  (COMMAND "STYLE" "" "" "5" "1.5" "" "" "" "")
  (SFERMA1)
  (SETQ T1 (POLAR BTF PI (* 0.3 H))
    T2 (POLAR BTF (* PI 1.5) (* 0.3 H))
    T3 (POLAR BTF PI AS)
    T4 (POLAR T3 (* PI 0.5) BS)
    TK1 (POLAR BTF 0 (* NC LS))
    TK (POLAR TK1 (* PI 0.5) H)
  )
  (COMMAND "PLINE" T1 "W" 1 1 BTF T2 ""
    "PLINE" T4 "W" 2 2 TK ""
    "CIRCLE" T1 3 "CIRCLE" T2 3
    "CIRCLE" T4 3 "CIRCLE" TK 3)
  (SETQ T1 (POLAR BTF 0 (* N1 LS))
    T2 (POLAR T1 (* PI 0.5) H)
    T3 (POLAR T2 0 LS)
    T4 (POLAR T1 0 LS)
  )
  (SETQ A (/ (- BS H) (+ AS (* NC LS)))
    N1LS (* N1 LS)
    N11LS (* (+ N1 1) LS)
  )

```

```

I 1
BT BTF
DZ (/ (FLOAT L) NZ)
SINB (/ H (SQRT (+ (* H H) (* LS LS))))
)
(SETQ I1 N1
D (* 0.1 H)
TT1 (ITOA (- (* 2 N) N1))
TT2 (ITOA I1)
TT3 (ITOA (+ N1 1))
TT4 (ITOA (- (* 2 N) N1 1))
T1T (POLAR T1 (* PI 1.5) D)
T2T (POLAR T2 (* PI 0.5) D)
T3T (POLAR T3 (* PI 0.5) D)
T4T (POLAR T4 (* PI 1.5) D)
TTL " Линия влияния для стержня ")
(COMMAND "TEXT" T1T "0" TT1
"TEXT" T2T "0" TT2
"TEXT" T3T "0" TT3
"TEXT" T4T "0" TT4)
(REPEAT 4
(SETQ BT (POLAR BT (* PI 1.5) (* 0.7 H))
BTK (POLAR BT 0 L)
Z 0)
(COMMAND "PLINE" BT "W" 1 1 BTK "")
(SETQ BTT (POLAR BT (* PI 0.5) (* 0.3 H)))
(COND
((= I 1) (COMMAND "TEXT" BTT ""
(STRCAT TTL TT2 " - " TT3)))
((= I 3) (COMMAND "TEXT" BTT "" (IF (< N1 NC)
(STRCAT TTL TT1 " - " TT3)
(STRCAT TTL TT2 " - " TT4))))
((= I 2) (COMMAND "TEXT" BTT ""
(STRCAT TTL TT1 " - " TT4)))
((= I 4) (COMMAND "TEXT" BTT ""
(STRCAT TTL TT3 " - " TT4))))
(REPEAT NZ
(COND
((= I 1) (COMMAND "PLINE" T2 "W" 2 2 T3 ""))
((= I 2) (COMMAND "PLINE" T1 "W" 2 2 T4 ""))
((= I 3) (IF (< N1 NC)
(COMMAND "PLINE" T1 "W" 2 2 T3 "")
(COMMAND "PLINE" T2 "W" 2 2 T4 "")))
((= I 4) (COMMAND "PLINE" T3 "W" 2 2 T4 ""))
)
)

```



```
(SETQ Z (+ DZ Z )
      L2 (ENTLAST)
      BTP (POLAR BTF 0 Z)
      PG (/ Z (+ (* A NC LS) H))
      P 1
      PV (* PG A)
      PY (- P PV)
      PX PG
      P1 (/ (* P (- N1LS Z)) LS)
      P2 (/ (* P (- Z N1LS)))
)
(COMMAND "PLINE" BTP "W" 0.5 5
(POLAR BTP (* PI 0.5) (* 0.3 H)) "")
(SETQ L1 (ENTLAST))
(COND ((= I 1)
      (IF (< N1 NC)
          (IF (<= Z N1LS) (SETQ S (/ (- (- N1LS Z) (* PY N1LS)) H))
              (SETQ S (/ (- 0 (* PY N1LS)) H))
          )
      (IF (<= Z N11LS) (SETQ S 0) (SETQ S (/ (- Z N11LS) H))))
)
((= I 2)
 (IF (< N1 NC)
     (IF (<= Z N11LS) (SETQ S (/ (- (* PY N11LS) (- N11LS Z)) H))
         (SETQ S (/ (* PY N11LS) H))
     )
 (IF (<= Z N1LS) (SETQ S 0)
     (SETQ S (/ (- N1LS Z) H))))
)
((= I 3)
 (IF (< N1 NC)
     (IF (<= Z N1LS) (SETQ S (/ (- 1 PY) SINB))
         (IF (AND (> Z N1LS) (< Z N11LS))
             (SETQ S (/ (- P1 PY) SINB))
             (SETQ S (/ (- 0 PY) SINB)))
     )
 (IF (<= Z N11LS) (SETQ S 0)
     (IF (AND (> Z N1LS) (< Z N11LS))
         (SETQ S (/ P1 SINB))
         (SETQ S (/ 1 SINB))))
)
((= I 4)
 (IF (< N1 NC)
     (IF (<= Z N11LS) (SETQ S (- PY 1)) (SETQ S PY))
 )
)
```

```

      (IF (< Z N11LS) (SETQ S 0) (SETQ S 1)))
    )
    (SETQ BT1 (POLAR BT 0 Z)
      S (* S (* 0.3 H))
      BT2 (POLAR BT1 (* PI 0.5) S))
    (COMMAND "PLINE" BT1 "W" 2 2 BT2 "")
    (ENTDEL L2)
    (ENTDEL L1)
  )
  (SETQ I (+ I 1))
  (SETQ ZN (RTOS (/ S (* 0.3 H)) 2 2))
  (COMMAND "TEXT" BT2 "0" ZN)
)
(IF (< N1 NC) (COMMAND "PLINE" T2 "W" 2 2 T3 T4 T1 T3 "")
  (COMMAND "PLINE" T1 "W" 2 2 T4 T3 T2 T4 ""))
(PRIN1)
)
(STRELA 40 200 320.0 62.0 8 4 220 20 2 32) ; вызов функции

```

Результат выполнения функции (STRELA...) представлен на рис. 6.16.

6.3. Расчет и параметрическое изображение различных способов выполнения производственного процесса

6.3.1. Постановка задачи

Задан некоторый производственный процесс, который включает M операций. Известно время выполнения каждой операции T_{ni} , число рабочих мест на каждой операции C_i и число изделий, которые необходимо изготовить с использованием данного процесса, $N = 15$. Кроме того, известны размер передаточной партии, $P = 5$ и время, необходимое на передачу партии изделий с одной операции на другую, $T_{mo} = 3$. Требуется определить длительность производственного процесса при различных способах выполнения задания и изобразить весь производственный процесс во времени с разбивкой по операциям, а в случае необходимости – и по передаточным партиям. Необходимая исходная информация представлена в табл. 6.1.

6.3.2. Выявление основных особенностей, взаимосвязей и количественных закономерностей

Известны три способа выполнения производственного процесса: последовательный, последовательно-параллельный, параллельный.

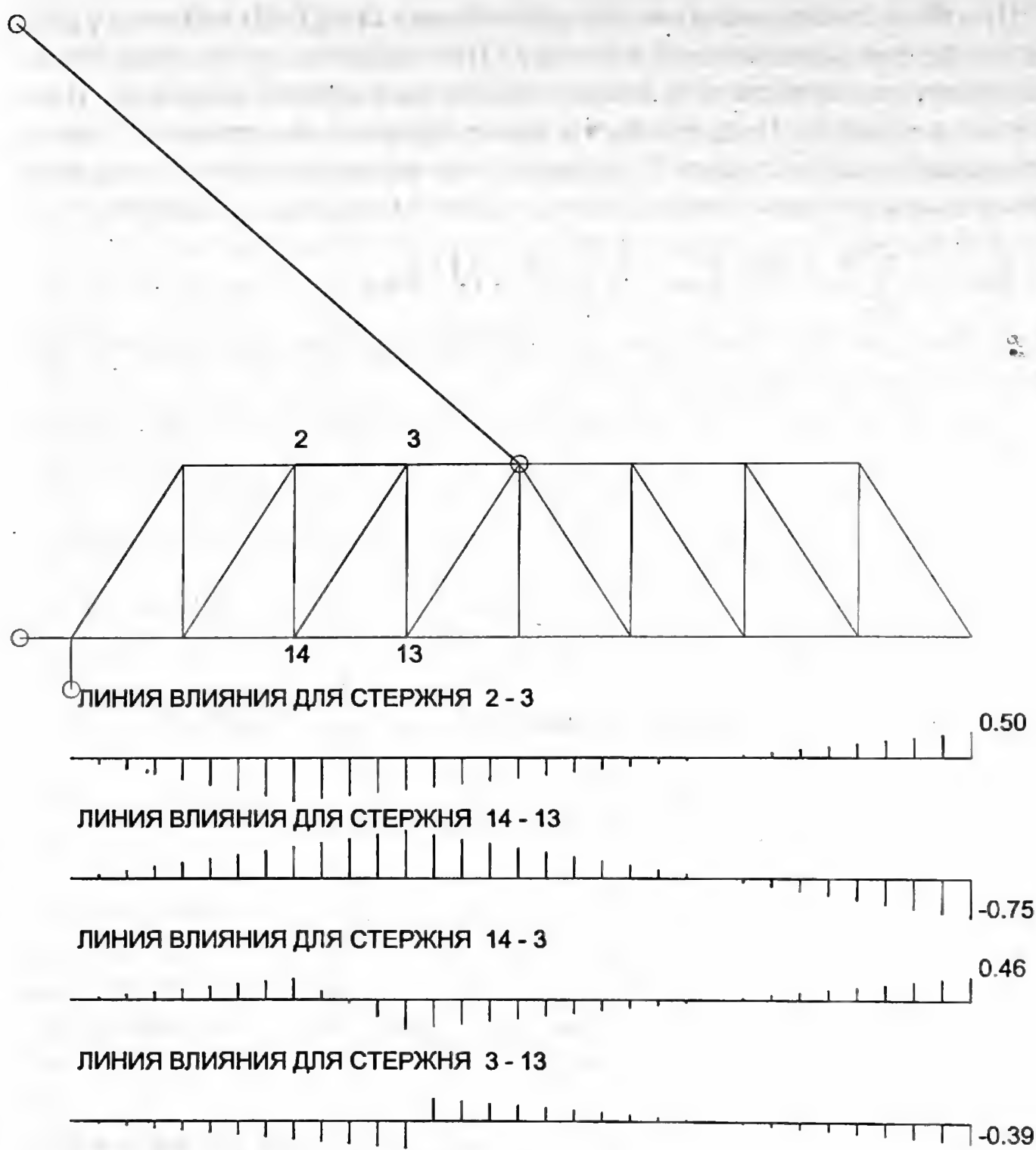


Рис. 6.16. Результаты расчета секции стрелы

Таблица 6.1

№ Операции	Норма времени	Число рабочих
1	4	2
2	3	1
3	4,5	3

При *последовательном способе* выполнения производственного процесса вся партия передается на последующую операцию лишь после полного окончания изготовления всех изделий на предыдущей операции. В этом случае длительность производственного процесса определяется как сумма операционных циклов T_{oi} и времени на межоперационные перерывы, связанные с передачей всех изделий с одной операции на другую:

$$T_{\text{посл}} = \sum_{i=1}^M T_{oi} + M * T_{\text{МО}} = N \sum_{i=1}^M \frac{T_{Hi}}{C_i} + M * T_{\text{МО}}$$

График последовательного способа выполнения производственного процесса представлен на рис. 6.17.

Последовательно-параллельный способ выполнения производственного процесса предполагает передачу изделий с операции на операцию передаточными партиями по P штук. При этом передача осуществляется таким образом, чтобы на каждой операции все N изделий изготавливались без перерыва.

За счет передачи изделий передаточными партиями можно частично совместить операции, а следовательно, сократить цикл изготовления всех изделий. В такой ситуации возможны два варианта:

а) предыдущий операционный цикл меньше последующего:

$$T_i < T_{i+1}$$

б) предыдущий операционный цикл больше последующего или равен ему:

$$T_i \geq T_{i+1}$$

В первом случае, когда $T_i < T_{i+1}$, начало изготовления изделий на последующей операции возможно сразу же после окончания изготовления на предыдущей операции первой передаточной партии. Следовательно, время совмещения двух смежных операций T_i и T_{i+1} определяется по времени с меньшим операционным циклом:

NN TN C

Длительность операций процесса $N = 30$, $T_{\text{МО}} = 6.00$

1 4.00 2.00

2 3.00 1.00

3 4.50 3.00

Длительность последовательного производственного процесса $TPP = 207.00$

Рис. 6.17. График последовательного способа выполнения производственного процесса

$$T_{\text{совм}, i+1} = (N - P) T_{\text{н}i} / C_i \text{ при } T_i < T_{i+1}$$

Во втором случае, когда $T_i \geq T_{i+1}$, начало изготовления изделий на последующей операции определяется из условия, что последняя передаточная партия, изготовленная на предыдущей операции, сразу же передается для изготовления на последующую операцию. Остальные передаточные партии должны быть переданы на последующую операцию с таким расчетом, чтобы на последующей операции процесс изготовления всех изделий был непрерывным. Время совмещения двух смежных операций T_i и T_{i+1} определяется по времени T_{i+1} , то есть операции с меньшим операционным циклом:

$$T_{\text{совм}, i+1} = (N - P) T_{\text{н}i+1} / C_{i+1} \text{ при } T_i \geq T_{i+1}$$

Из рассмотренных случаев видно, что экономия времени при изготовлении изделий определяется по времени меньшего операционного цикла из двух смежных. Это можно записать следующим образом:

$$T_{\text{совм}, i+1} = (N - P) \{T_{\text{н}i} / C_i, T_{\text{н}i+1} / C_{i+1}\}_{\text{MIN}}$$

Время всего производственного процесса с учетом перерывов на передачу изделий с операции на операцию составит:

$$T_{\text{пол. пар}} = T_{\text{пол.}} - \sum_{i=1}^{M-1} T_{\text{совм}, i+1} = N \sum_{i=1}^M \frac{T_{\text{н}i}}{C_i} + M * T_{\text{МО}} - (N - P) \sum_{i=1}^{M-1} \left\{ \frac{T_{\text{н}i}}{C_i}, \frac{T_{\text{н}i+1}}{C_{i+1}} \right\}_{\text{MIN}}$$

График последовательно-параллельного способа организации выполнения производственного процесса представлен на рис. 6.18.

При *параллельном способе* выполнения производственного процесса каждая передаточная партия сразу после ее изготовления передается для изготовления на следующую операцию. Начало изготовления второй и последующей передаточных партий определяется по самой продолжительной операции. Таким образом, зная время изготовления передаточной партии на самой продолжительной операции и откладывая это время на графике для первой передаточной операции, можно определить время изготовления соответствующих передаточных партий. В результате операция, которая характеризуется максимальной продолжительностью изготовления изделий,

NN TN C Длительность операций процесса N = 30, TMO = 6.00, P = 10.00

1 4.00 2.00

2 3.00 1.00

3 4.50 3.00

Длительность последовательно-параллельного производственного процесса TRP = 137.00

Рис. 6.18. График последовательно-параллельного способа производственного процесса

NN TN C Длительность операций процесса N = 30, TMO = 6.00, P = 10.00

1 4.00 2.00

2 3.00 1.00

3 4.50 3.00

Длительность последовательно-параллельного производственного процесса TPR = 137.00

Рис. 6.19. График параллельного способа выполнения производственного процесса не будет иметь перерывов между передаточными партиями. Время выполнения производственного процесса при параллельном способе организации производства составит:

$$T_{\text{пар.}} = P \sum_{i=1}^M \frac{T_{Hi}}{C_i} + M * T_{MO} + (N - P) \left\{ \frac{T_{H1}}{C_1}, \frac{T_{H2}}{C_2}, \dots, \frac{T_{Hi}}{C_i} \right\}_{\text{MAX}}$$

Выражение в фигурных скобках означает, что из всего множества отношений выбирается одно, имеющее максимальное значение.

График параллельного способа выполнения производственного процесса представлен на рис. 6.19.

6.3.3. Разработка последовательности действий и комплекса функций расчета и параметрического изображения различных способов выполнения производственного процесса

Порядок расчета и создания параметрического изображения различных способов выполнения производственного процесса включает два основных этапа:

- *определение местоположения заголовка, его отображение и определение базовой точки для прорисовки графика.* Допустим, начало заголовка находится в точке ВТ с координатами: X = 10, а Y = 180. Начало прорисовки графика определяется точкой ВТО, которая смещена по отношению к точке ВТ на 50 единиц вправо. Координаты точек ВТ и ВТО придется определять для каждой операции путем смещения их на 20 единиц вниз. Следовательно направление (угол) перемещения достаточно определить один раз. В AutoLISP отсчет угла идет от горизонтальной линии против часовой стрелки, перемещение вниз равносильно перемещению под углом 270°. Обозначим эту величину PIS. Здесь же определим ряд дополнительных переменных;

- вывод на чертеж входных данных, расчет и отображение времени выполнения операционного цикла. Данный этап целесообразно выполнять в цикле. Число обращений равно числу операций в производственном процессе. В каждом цикле вводится номер операции C_i . При этом базовые точки для ввода исходных данных ВТ и построения графика ВТО автоматически пересчитываются и перемещаются на 20 единиц вниз.

Расчет и отображение времени выполнения операции зависит от способа выполнения производственного процесса.

При последовательном способе длительность выполнения каждой операции определяется по формуле:

$$T_i = N T_{ii} / C_i$$

При этом начальная точка отображения времени выполнения следующей операции определяется по конечной точке предыдущей со смещением по горизонтали на время межоперационных перерывов T_{mo} и вниз на величину заданного шага – (20 единиц). Ширина линии для прорисовки времени выполнения операции принята равной двум единицам.

При последовательно-параллельном способе длительность выполнения операции остается такой же, как и при последовательном, однако в смежных операциях допускается совмещение их выполнения.

Если операционный цикл T_i меньше операционного цикла T_{i+1} , то производится смещение начальной точки операции ВТО на величину $(N - P) T_{ii} / C_i$ влево и на 20 единиц вниз относительно конечной точки ВТК, относящейся к предшествующей операции.

При параллельном способе предварительно определяется самая продолжительная операция в цикле, затем – длительность выполнения передаточной партии $T_{pi} = P T_{ii} / C_i$. Начало выполнения каждой последующей передаточной партии смещено вправо относительно начала выполнения предыдущей передаточной партии на величину, которая равна времени изготовления передаточной партии на самой продолжительной операции. В этом случае операция, которая характеризуется самым длительным временем выполнения, не имеет перерывов.

Порядок определения местоположения заголовка, его изображения и определения базовой точки для прорисовки графика при последовательном способе состоит в следующем:

```
(SETQ I 1
  ВТ '(10 180) ; базовая точка ввода данных
  ВТО '(100 180)) ; базовая точка для прорисовки графика
(COMMAND "TEXT" ВТ "0" (STRCAT " NN TN C "
  " Длительность операций процесса N = " (ITOA N)
  " TMO = " (RTOS TMO 2 2)))
```

В функции (**COMMAND...**) используется одна команда графического редактора **"TEXT"** для вывода на чертеж текстовых данных и три функции AutoLISP: (**STRCAT...**), (**ITOA N**), (**RTOS TMO 2 2**).

Функция (**STRCAT...**) объединяет, сцепляет все строковые константы в одну (один текст).

Функция (**ITOA N**) преобразует целое число (число изделий **N**) в строковую константу для последующего вывода на чертеж.

Функция (**RTOS TMO 2 2**) преобразует действительное число (время межоперационных перерывов **TMO**) в строковую константу, чтобы затем вывести ее на чертеж. Второй аргумент функции – 2 указывает, что число должно быть представлено в десятичном виде. Третий аргумент – 2 указывает, что число знаков после запятой должно быть равно двум.

Вывод на чертеж входных данных, расчет и отображение времени выполнения операционного цикла может выглядеть следующим образом:

```
(FOREACH EL LTC
  (SETQ TN (CAR EL)      ; норма времени на операцию
    C (CADR EL)         ; число рабочих мест на операции
    BT (POLAR BT (* PI 1.5) 20)
    BTO (POLAR BTO (* PI 1.5) 20))
  (COMMAND "TEXT" BT "0" (STRCAT " " (ITOA I) " "
    (RTOS TN 2 2) " " (RTOS C 2 2) " "))
  (SETQ TO (/ (* N TN ) C)
    BTK (POLAR BTO 0 TO))
  (COMMAND "PLINE" BTO "W" 2 2 BTK "")
  (SETQ BTO (POLAR BTK 0 TMO)
    I (+ I 1))
)
```

Функция (**FOREACH EL LTC...**) последовательно извлекает из списка **LTC** элементы **EL**, которые, в свою очередь, представляют список из двух элементов (**T C**) – нормы времени на операцию **T** и числа рабочих мест на операции **C**. При помощи функций (**NTH 0 EL**) и (**NTH 1 EL**) из списка извлекаются элементы **T** и **C**. Значение 0 в функции (**NTH 0 EL**) означает извлечение из списка первого элемента, значение 1 – второго элемента списка и т. д. Затем определяются базовые точки **BT** и **BTO** для ввода исходных данных и создания графического изображения. Исходные данные вводятся с помощью функции (**COMMAND...**). Далее определяется длительность операционного цикла **TO**, координаты конечной точки **BTO** и с помощью команды **"PLINE"** отображается линия, а также осуществляется подготовка к следующему циклу.

В заключение определяется длительность последовательного производственного процесса:

```
(SETQ BTO (POLAR BT 0 90)
      TPP (DISTANCE BTO BTK))
(COMMAND "TEXT" (POLAR BT (* PI 1.5) 20) "0"
(STRCAT "Длительность последовательного производственного "
        "процесса TPP = " (RTOS TPP 2 2)))
```

Точка ВТО определяет начало производственного цикла, функция (**DISTANCE BTT BTK**) – время выполнения всего производственного цикла: от начальной точки ВТО до конечной точки ВТК на последней операции. Команда "TEXT" включает в чертеж текстовую информацию, начиная с точки (**POLAR BT (* PI 1.5) 20**) с углом наклона текста 0. Функция (STRCAT...) формирует текстовую строку из текста и числа, которое преобразуется в текстовый вид с помощью функции (RTOS TPP 2 2).

6.3.4. Функции расчета и параметрического изображения различных способов выполнения производственного процесса

Ниже представлена функция (**POSL LTC N TMO**).

```
; *****
; Функция (POSL LTC N TMO) обеспечивает расчет и параметрическое
; изображение последовательного способа выполнения любого
; производственного процесса
; А р г у м е н т ы  ф у н к ц и и:
; LTC - список норм времени T_ и числа рабочих C_ в процессе
; вида ((T1 C1) (T2 C2) (T3 C3) . . .)
; N - число изготавливаемых изделий
; TMO - время межоперационных перерывов
; Пример использования функции (POSL LTC N TMO)
; (POSL '((4 2) (3 1) (4.5 3)) 30 6)
; *****
(DEFUN POSL (LTC N TMO)
  (SETVAR "CMDECHO" 0)
  (SETVAR "BLIPMODE" 0)
  (COMMAND "LIMITS" "0,0" "297,210")
  (COMMAND "STYLE" "" "" "4" "1" "" "" "" "")
  (SETQ I 1
        BT '(10 180) ; базовая точка ввода данных
        BTO '(100 180)) ; базовая точка для прорисовки графика
  (COMMAND "TEXT" BT "0" (STRCAT " NN TN C "
    " Длительность операций процесса N = " (ITOA N)
    ", TMO = " (RTOS TMO 2 2)))
  (FOREACH EL LTC
```

```

(SETQ TN (CAR EL)           ; норма времени на операцию
  C (CADR EL)               ; число рабочих мест на операции
  BT (POLAR BT (* PI 1.5) 20)
  BTO (POLAR BTO (* PI 1.5) 20))
(COMMAND "TEXT" BT "0" (STRCAT " " (ITOA I) " "
  (RTOS TN 2 2) " " (RTOS C 2 2) " "))
(SETQ TO (/ (* N TN) C)
  BTK (POLAR BTO 0 TO))
(COMMAND "PLINE" BTO "W" 2 2 BTK "")
(SETQ BTO (POLAR BTK 0 TMO)
  I (+ I 1)))
(SETQ BTO (POLAR BT 0 90)
  TPP (DISTANCE BTO BTK))
(COMMAND "TEXT" (POLAR BT (* PI 1.5) 20) "0"
  (STRCAT "Длительность последовательного производственного "
    "процесса TPP = " (RTOS TPP 2 2)))
(COMMAND "ZOOM" "ALL" "" )
)
(POSL '((4 2) (3 1) (4.5 3)) 30 6)           ; вызов функции

```

Результат выполнения функции (POSL...) представлен на рис. 6.20.

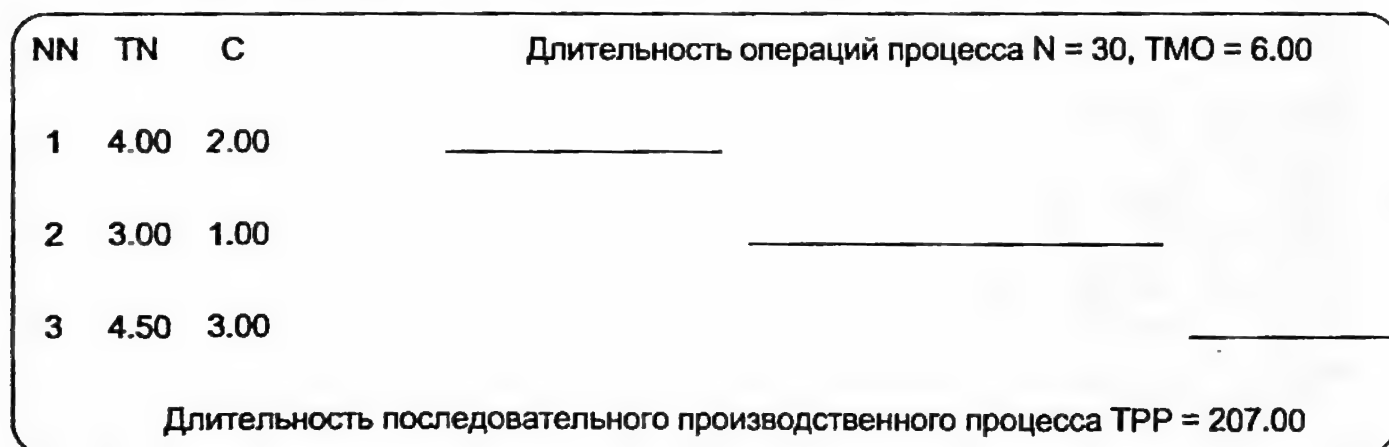


Рис. 6.20. Параметрическое изображение последовательного способа выполнения производственного процесса

Ниже представлена функция (PPAR LTC N P TMO).

```

; *****
; Функция (PPAR LTC N P TMO) обеспечивает расчет и параметрическое
; изображение последовательно-параллельного способа выполнения
; производственного процесса
; А р г у м е н т ы  ф у н к ц и и:
; LTC - список норм времени T_ и числа рабочих C_ на операциях
; вида ((T1 C1) (T2 C2) (T3 C3)...)
; N - число изготавливаемых изделий
; P - величина передаточной партии

```



```

; TMO - время межоперационных перерывов
; Пример использования функции (PPAR LTC N P TMO)
; (PPAR '((4 2) (3 1) (4.5 3)) 30 10 6)
; *****
(DEFUN PPAR (LTC N P TMO)
  (SETVAR "CMDECHO" 0)
  (SETVAR "BLIPMODE" 0)
  (COMMAND "LIMITS" "0,0" "297,210" )
  (COMMAND "STYLE" "" "" "4" "1" "" "" "" "")
  (SETQ I 1
    BT '(10 180)
    BTO '(100 180)
    TR 0
    TRN 0 )
  (COMMAND "TEXT" BT "0" (STRCAT " NN TN С Длительность"
    " операций процесса N = " (ITOA N) ", TMO = " (RTOS TMO 2 2)
    ", P = " (RTOS P 2 2)))
  (FOREACH EL LTC
    (SETQ TN (CAR EL)
      C (CADR EL) TC (/ TN C)
      BT (POLAR BT (* PI 1.5) 20)
      BTO (POLAR BTO (* PI 1.5) 20)
      BTT (POLAR BT 0 90))
    (COMMAND "TEXT" BT "0" (STRCAT " " (ITOA I) " "
      (RTOS TN 2 2) " " (RTOS C 2 2) " "))
    (SETQ TO (/ (* N TN ) C)
      TPN (/ (* (- N P) TN ) C))
    (IF (> TR TC) (IF (> I 1) (SETQ BTO (POLAR BTO PI TPN)))
      (SETQ BTO (POLAR BTO PI TRN)))
      (SETQ BTK (POLAR BTO 0 TO))
      (COMMAND "PLINE" BTO "W" 2 2 BTK "")
      (SETQ BTO (POLAR BTK 0 TMO)
        I (+ I 1))
      (SETQ TR TC
        TRN TPN) )
      (SETQ TPP (DISTANCE BTT BTK))
      (COMMAND "TEXT" (POLAR BT (* PI 1.5) 20) "0"
        (STRCAT "Длительность последовательно-параллельного "
          "производственного процесса TPP = " (RTOS TPP 2 2)))
      (COMMAND "ZOOM" "ALL" "" )
    )
  (PPAR '((4 2) (3 1) (4.5 3)) 30 10 6) ; вызов функции

```

Результат выполнения функции (PPAR...) представлен на рис. 6.21. Ниже представлена функция (PAR LTC N P TMO).

NN TN C Длительность операций процесса N = 30, TMO = 6.00, P = 10.00

1 4.00 2.00

2 3.00 1.00

3 4.50 3.00

Длительность последовательно-параллельного производственного процесса TPP = 137.00

Рис. 6.21. Параметрическое изображение последовательно-параллельного способа выполнения производственного процесса

```
; *****
; Функция (PAR LTC N P TMO) обеспечивает расчет и параметрическое
; представление параллельного способа выполнения
; производственного процесса
; А р г у м е н т ы  ф у н к ц и и:
; LTC - список норм времени T_ и числа рабочих C_ на
; операциях вида ((T1 C1) (T2 C2) (T3 C3)...)
; N - число изготавливаемых изделий
; P - величина передаточной партии
; TMO - время межоперационных перерывов
; Пример использования функции (PAR LTC N P TMO)
; (PAR '((4 2) (3 1) (4.5 3)) 30 10 6)
; *****
(DEFUN PAR (LTC N P TMO)
  (SETVAR "CMDECHO" 0) (SETVAR "BLIPMODE" 0)
  (COMMAND "LIMITS" "0,0" "297,210")
  (COMMAND "STYLE" "" "" "4" "1" "" "" "" "")
  (SETQ I 1
    BT '(10 180)
    BTO '(100 180)
    NP (/ N P)
    TCMAX 0)
  (COMMAND "TEXT" BT "0" (STRCAT " NN TN C "
    " Длительность операций процесса N = " (ITOA N)
    ", TMO = " (RTOS TMO 2 2) ", P = " (RTOS P 2 2)))
  (FOREACH EL LTC
    (SETQ TN (CAR EL)
      C (CADR EL)
      TC (/ TN C))
    (IF (> TC TCMAX) (SETQ TCMAX TC)))
```

```

(SETQ TPMAX (* TCMAH P))
(FOREACH EL LTC
  (SETQ TN (CAR EL)
    C (CADR EL)
    BT (POLAR BT (* PI 1.5) 20)
    BTO (POLAR BTO (* PI 1.5) 20)
    BTT (POLAR BT 0 90))
  (COMMAND "TEXT" BT "0" (STRCAT " " (ITOA I) " "
    (RTOS TN 2 2) " " (RTOS C 2 2) " ")))
(SETQ TO (/ (* P TN) C)
  BTK (POLAR BTO 0 TO)
  T1 BTO
  T2 BTK)
(REPEAT NP
  (SETQ BT2 T2)
  (COMMAND "PLINE" T1 "W" 2 2 T2 ""))
(SETQ T1 (POLAR T1 0 TPMAX)
  T2 (POLAR T1 0 TO))
(SETQ BTO (POLAR BTK 0 TMO)
  I (+ I 1))
)
(SETQ TPP (DISTANCE BTT BT2))
(COMMAND "TEXT" (POLAR BT (* PI 1.5) 20) "0")
(STRCAT "Длительность параллельного производственного "
  "процесса TPP = " (RTOS TPP 2 2)))
(COMMAND "ZOOM" "ALL" ""))
(PAR '((4 2) (3 1) (4.5 3)) 30 10 6) ; вызов функции

```

Результат выполнения функции (**PAR...**) представлен на рис. 6.22.

Часто возникают ситуации, когда график изображения производственного процесса выходит за рамки экрана. В этом случае возможны два способа просмотра графика выполнения всего производственного процесса.

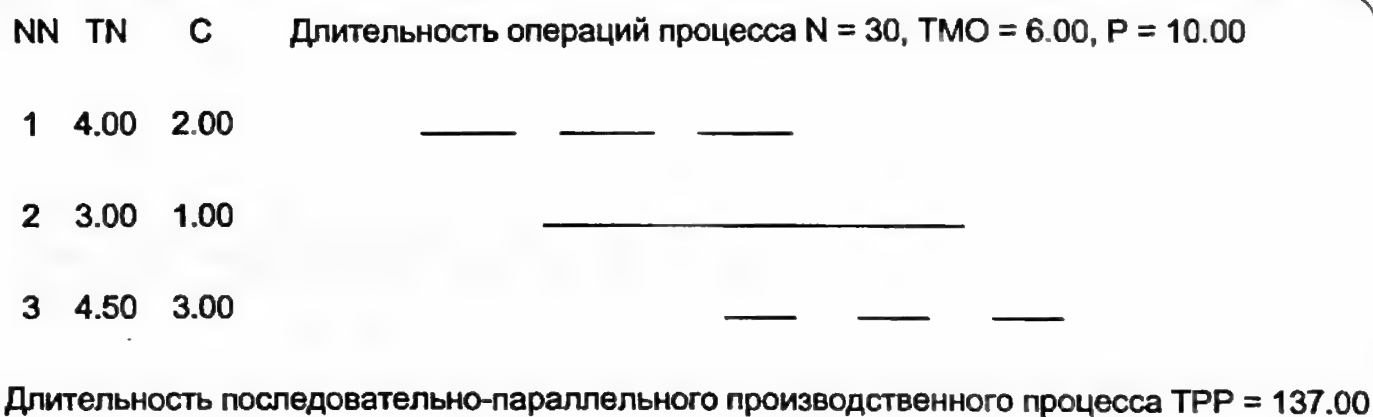


Рис. 6.22. Параметрическое изображение параллельного способа выполнения производственного процесса

Первый, самый простой, но не самый лучший, – использовать команду графического редактора "ZOOM" (Показать) с ключом A (ALL – Все). График производственного процесса появится на экране полностью, однако подписи на графике могут быть очень мелкими.

Второй способ связан с дополнительными операциями вычислений и обеспечивает автоматическое уменьшение только графика. Для того чтобы воспользоваться вторым способом, достаточно определить коэффициент масштабирования КМ для самого продолжительного производственного процесса – последовательного, который определяется следующим образом:

```
(SETQ TD 0)
(Foreach EL LTC ; определение длительности процесса
  (SETQ T (NTH 0 EL) ; норма времени на операцию
    C (NTH 1 EL) ; число рабочих мест на операции
    TO (/ (* N TN ) C) ; длительность выполнения операции
    TD (+ TD TO)) ; длительность выполнения процесса
)
(SETQ KM (/ 200.0 TD)) ; определение коэффициента масштабирования
```

Используя коэффициент масштабирования КМ, необходимо ввести соответствующие изменения, связанные с определением времени выполнения операций ТО, времени межоперационных перерывов ТМО и всего производственного процесса ТРР.

После определения длительности процесса ТО:

```
(SETQ TO (/ (* N TN ) C))
```

следует скорректировать величину длительности процесса:

```
(SETQ TO (* TO KM))
```

Перед использованием времени межоперационного цикла ТМО необходимо задать его скорректированное значение:

```
(SETQ TMO (* TMO KM))
```

Поскольку ТО и ТМО были скорректированы с использованием коэффициента масштабирования КМ, необходимо восстановить реальную длительность всего процесса ТРР.

Там, где определены значения ТРР

```
(SETQ TRP (DISTANCE BTO BTK))
```

их следует заменить на выражение для определения реального времени ТРР:

```
(SETQ TRP (/ TRP KM))
```

Одновременно с модернизацией функций введем новое условие. Допустим, время межоперационных перерывов непостоянно. В таком случае, кроме $T_{_}$ – нормы времени на выполнение операции и $C_{_}$ – числа

рабочих мест на операции, в список LTC следует ввести время меж-операционных перерывов ТМО:

((T1 C1 ТМО1) (T2 C2 ТМО2) (T3 C3 ТМО3)...))

В окончательном виде функция (POSL1 LTC N) будет выглядеть следующим образом:

```
; *****
; Функция (POSL1 LTC N) обеспечивает расчет и параметрическое
; изображение последовательного способа выполнения
; производственного процесса
; А р г у м е н т ы   ф у н к ц и и:
; LTC - список норм времени T_, числа рабочих мест C_ и времени
; межоперационных перерывов ТМО_ в производственном процессе
; вида ((T1 C1 ТМО1) (T2 C2 ТМО2) (T3 C3 ТМО3)...)
; N - число изготавливаемых изделий
; Пример использования функции (POSL1 LTC N)
; (POSL1 ' ((4 2 0.5) (3 1 1.0) (4.5 3 1.5)) 30)
; *****
(DEFUN POSL1 (LTC N)
  (SETVAR "CMDECHO" 0)
  (SETVAR "BLIPMODE" 0)
  (COMMAND "LIMITS" "0,0" "297,210")
  (COMMAND "STYLE" "" "" "4" "1" "" "" "" "")
  (SETQ I 1
    BT '(10 180) ; базовая точка ввода данных
    BTO '(100 180)) ; базовая точка для прорисовки графика
  (COMMAND "TEXT" BT "0" (STRCAT " NN TN C ТМО "
    " Длительность операций процесса при N = " (ITOA N)))
  (SETQ TD 0)
  (FOREACH EL LTC ; определение длительности процесса
    (SETQ TN (NTH 0 EL) ; норма времени на операцию
      C (NTH 1 EL) ; число рабочих мест на операции
      TO (/ (* N TN ) C) ; длительность выполнения операции
      TD (+ TD TO)) ; длительность выполнения процесса
    )
  (SETQ KM (/ 200.0 TD)) ; определение коэффициента масштабирования
  (FOREACH EL LTC
    (SETQ TN (NTH 0 EL) ; норма времени на операцию
      C (NTH 1 EL) ; число рабочих мест на операции
      ТМО (NTH 2 EL) ; длительность межоперационного
        ; перерыва
      BT (POLAR BT (* PI 1.5) 20) ; точка ввода данных
      BTO (POLAR BTO (* PI 1.5) 20) ; точка начала операции
      TO (/ (* N TN ) C) ; длительность выполнения операции
```



```

      TO (* TO KM)           ; скорректированная длительность операций
      BTK (POLAR BTO 0 TO))   ; точка окончания операции
(COMMAND "TEXT" BT "0" (STRCAT " " (ITOA I) " "
(RTOS TN 2 2) " " (RTOS C 2 2) " " (RTOS TMO 2 2) " "))
(COMMAND "PLINE" BTO "W" 2 2 BTK " ")
(SETQ TMO (* TMO KM)         ; скорость время межоперационных перерывов
BTO (POLAR BTK 0 TMO)        ; точка начала следующей операции
I (+ I 1))
)
(SETQ BTO (POLAR BT 0 90)    ; точка начала процесса
      TPP (DISTANCE BTO BTK) ; скорректир. длительность процесса
      TPP (/ TPP KM))        ; реальная длительность процесса
(COMMAND "TEXT" (POLAR BT (* PI 1.5) 20) "0"
(STRCAT "Длительность последовательного производственного"
" процесса TPP = " (RTOS TPP 2 2))
)
(COMMAND "ZOOM" "ALL" "" )
)
(POSL1 '((4 2 0.5) (3 1 1.0) (4.5 3 0.0)) 60) ; вызов функции

```

Результат выполнения функции (**POSL1...**) представлен на рис. 6.23.

NN TN C TMO Длительность операций процесса при N = 60

1 4.00 2.00 0.50

2 3.00 1.00 1.00

3 4.50 3.00 0.00

Длительность последовательного производственного процесса TPP = 391.50

Рис. 6.23. Параметрическое изображение последовательного способа выполнения производственного процесса

Аналогичную модернизацию можно осуществить и для функций (**PPAR...**) и (**PAR...**).

```

; *****
; Функция (PPAR1 LTC N P) обеспечивает расчет и параметрическое
; изображение последовательно-параллельного способа выполнения
; производственного процесса
; А р г у м е н т ы   ф у н к ц и и:
; LTC - список норм времени T_ ; числа рабочих мест C_ на
; операциях и времени межоперационных перерывов TMO_ вида
; ((T1 C1 TMO1) (T2 C2 TMO2) (T3 C3 TMO3)...)
; N - число изготавливаемых изделий

```

```

; P - величина передаточной партии
; Пример использования функции (PPAR1 LTC N P)
; (PPAR1 ' ((4 2 0.5) (3 1 1.0) (4.5 3 1.5)) 30 10)
; *****
(DEFUN PPAR1 (LTC N P)
  (SETVAR "CMDECHO" 0)
  (SETVAR "BLIPMODE" 0)
  (COMMAND "LIMITS" "0,0" "297,210")
  (COMMAND "STYLE" "" "" "4" "1" "" "" "" "")
  (SETQ I 1
    BT '(10 180)
    BTO '(100 180)
    TR 0
    TRN 0 )
  (COMMAND "TEXT" BT "0" (STRCAT " NN TN C TMO "
    " Длительность операций процесса при N = " (ITOA N)
    " и P = " (ITOA P)))
  (SETQ TD 0)
  (FOREACH EL LTC ; определение длительности процесса
    (SETQ TN (NTH 0 EL) ; норма времени на операцию
      C (NTH 1 EL) ; число рабочих мест на операции
      TO (/ (* N TN ) C) ; длительность выполнения операции
      TD (+ TD TO))) ; длительность выполнения процесса
  (SETQ KM (/ 200.0 TD) ; определение коэффициента масштабирования
  (FOREACH EL LTC
    (SETQ TN (NTH 0 EL)
      C (NTH 1 EL)
      TC (/ TN C)
      TMO (NTH 2 EL)
      BT (POLAR BT (* PI 1.5) 20)
      BTO (POLAR BTO (* PI 1.5) 20)
      BTT (POLAR BT 0 90))
    (COMMAND "TEXT" BT "0" (STRCAT " " (ITOA I) " "
      (RTOS TN 2 2) " " (RTOS C 2 2) " " (RTOS TMO 2 2) " ")))
    (SETQ TO (/ (* N TN ) C)
      TO (* TO KM)
      TPN (/ (* (- N P) TN ) C)
      TPN (* TPN KM))
  (IF (> TR TC) (IF (> I 1) (SETQ BTO (POLAR BTO PI TPN)))
  (SETQ BTO (POLAR BTO PI TRN)))
  (SETQ BTK (POLAR BTO 0 TO))
  (COMMAND "PLINE" BTO "W" 2 2 BTK "")
  (SETQ BTO (POLAR BTK 0 TMO)
    I (+ I 1))

```

```

(SETQ TR TC TRN TPN))
(SETQ TPP (DISTANCE BTT BTK)
      TPP (/ TPP KM))
(COMMAND "TEXT" (POLAR BT (* PI 1.5) 20) "0"
(STRCAT "Длительность последовательно-параллельного производственного "
"процесса TPP = " (RTOS TPP 2 2)))
(COMMAND "ZOOM" "ALL" "" ))
(PPAR1 '((4 2 0.5) (3 1 1.0) (4.5 3 1.5)) 60 10)

```

Результат выполнения функции (PPAR1...) представлен на рис. 6.24.

NN TN C TMO Длительность операций процесса при N = 60 и P = 10

1 4.00 2.00 0.50

2 3.00 1.00 1.00

3 4.50 3.00 1.50

Длит. последовательно-параллельного производственного процесса TPP = 217.93

Рис. 6.24. Параметрическое изображение последовательно-параллельного способа выполнения производственного процесса

```

; *****
; Функция (PAR1 LTC N P) обеспечивает расчет и параметрическое
; изображение параллельного способа выполнения
; производственного процесса
; А р г у м е н т ы  ф у н к ц и и:
; LTC - список норм времени T_, числа рабочих мест C_ и времени
; междооперационных перерывов TMO_ в производственном процессе
; вида ((T1 C1 TMO1) (T2 C2 TMO2) (T3 C3 TMO3)...)
; N - число изготавливаемых изделий
; P - величина передаточной партии
; Пример использования функции (PAR1 LTC N P)
; (PAR1 '((4 2 0.5) (3 1 1.0) (4.5 3 1.5)) 30 10)
; *****
(DEFUN PAR1 (LTC N P)
  (SETVAR "CMDECHO" 0)
  (SETVAR "BLIPMODE" 0)
  (COMMAND "LIMITS" "0,0" "297,210" )
  (COMMAND "STYLE" "" "" "4" "1" "" "" "" "" )
  (SETQ I 1
    BT '(10 180) ; базовая точка ввода данных
    BTO '(100 180) ; базовая точка для прорисовки графика
    NP (/ N P) ; число передаточных партий

```

```

TCMAX 0)
(COMMAND "TEXT" BT "0" (STRCAT " NN TN C TMO "
" Длительность операций процесса при N = " (ITOA N)
" и P = " (ITOA P)))
(SETQ TD 0)
(Foreach EL LTC ; определение длит. послед. процесса
(SETQ TN (NTH 0 EL) ; норма времени на операцию
C (NTH 1 EL) ; число рабочих мест на операции
TO (/ (* N TN ) C) ; длительность выполнения операции
TD (+ TD TO)) ; длительность выполнения процесса
)
(SETQ KM (/ 200.0 TD)) ; определение коэффициента масштабирования
(Foreach EL LTC
(SETQ TN (NTH 0 EL)
C (NTH 1 EL)
TC (/ TN C)
TC (* TC KM))
(IF (> TC TCMAX) (SETQ TCMAX TC))
)
(SETQ TPMAX (* TCMAX P))
(Foreach EL LTC
(SETQ TN (NTH 0 EL)
C (NTH 1 EL)
TMO (NTH 2 EL)
BT (POLAR BT (* PI 1.5) 20) ; точка ввода данных
BTO (POLAR BTO (* PI 1.5) 20) ; точка начала партии
BTT (POLAR BT 0 90)
TO (/ (* P TN ) C) ; длительность изготовления партии
TO (* TO KM) ; скорректированная длительность
BTK (POLAR BTO 0 TO) ; точка окончания изготовления партии
T1 BTO
T2 BTK)
(COMMAND "TEXT" BT "0" (STRCAT " " (ITOA I) " "
(RTOS TN 2 2) " " (RTOS C 2 2) " " (RTOS TMO 2 2) " ")))
(REPEAT NP
(SETQ BT2 T2)
(COMMAND "PLINE" T1 "W" 2 2 T2 "")
(SETQ T1 (POLAR T1 0 TPMAX)
T2 (POLAR T1 0 TO))
)
(SETQ TMO (* TMO KM) ; скорр. время межоперационных перерывов
BTO (POLAR BTK 0 TMO)
I (+ I 1))
)
(SETQ TPP (DISTANCE BTT BT2) ; скоррект. длительность процесса
TPP (/ TPP KM)) ; реальная длительность процесса

```

```
(COMMAND "TEXT" (POLAR BT (* PI 1.5) 20) "0"  
(STRCAT " Длительность параллельного "  
  "производственного процесса TPP = " (RTOS TPP 2 2)))  
(COMMAND "ZOOM" "ALL" "" ))  
(PAR1 ' ((4 2 0.5) (3 1 1.0 ) (4.5 3 0.0)) 60 10)
```

Результат выполнения функции (PAR1...) представлен на рис. 6.25.

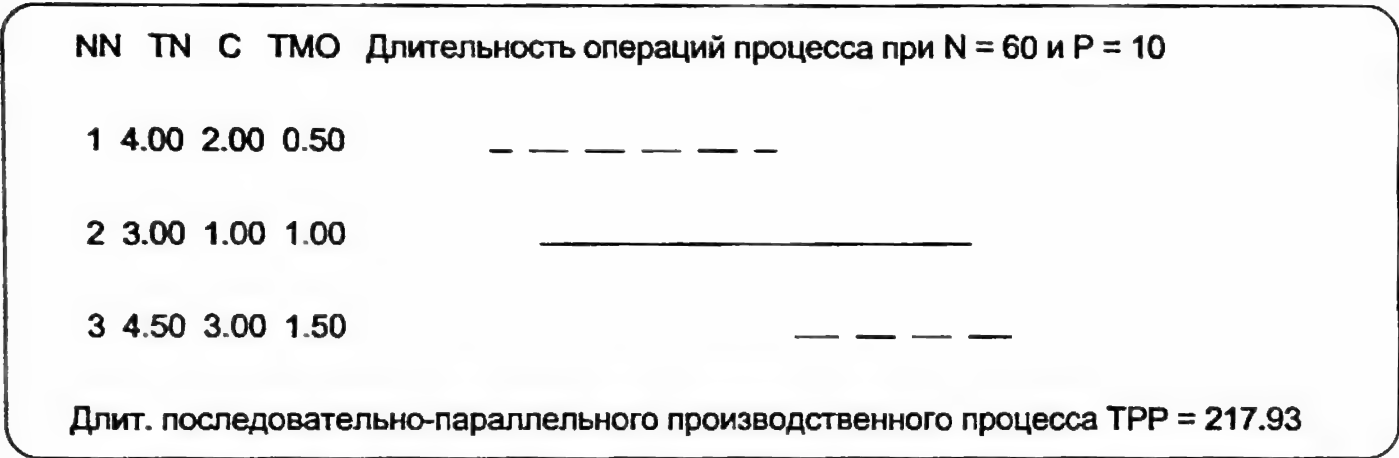


Рис. 6.25. Параметреское изображение параллельного способа выполнения производственного процесса

6.4. Оптимизация загрузки оборудования

6.4.1. Постановка задачи

- Необходимо организовать производство нескольких видов изделий, каждое из которых должно быть обработано на двух станках (оборудовании...). Известно время и последовательность обработки каждого изделия на каждом станке (см. табл. 6.2). Требуется определить такую последовательность запуска изделий в производство, чтобы суммарное время на изготовление всех изделий на двух станках было минимальным.

6.4.2. Выявление основных особенностей, взаимосвязей и количественных закономерностей

Время обработки j-го изделия на i-ом станке обозначим T_{ij} . Предполагается, что время перехода изделия от одного станка к другому незначительно, им можно пренебречь или включить в длительность операции.

Таблица 6.2

$C_i \setminus I_j$	I_1	I_2	I_3	I_4	I_5	I_6
Станок 1	6	4	6	5	7	4
Станок 2	5	2*	3	6	6	7

Каждое изделие обрабатывается вначале на первом станке, затем на втором. Обработка изделия на первом станке должна быть завершена прежде, чем она начнется на втором.

Исходный порядок запуска изделий в производство, который, может быть, и не самый оптимальный, представим следующим образом:

$$I_1 \rightarrow I_2 \rightarrow I_3 \rightarrow I_4 \rightarrow I_5 \rightarrow I_6$$

Процесс изготовления изделий отобразим графически (см. рис. 6.26).

Критерий оптимизации представляет собой суммарное время обработки изделий на двух станках (T) и определяется как сумма времени простоя второго станка в ожидании изделий с первого станка и времени обработки этих изделий на втором станке:

$$T = \sum_{j=1}^M T_{2j} + \sum_{j=1}^M T_{пj} = (5 + 2 + 3 + 6 + 6 + 7) + (6 + 3 + 2 + 1) = 29 + 12 = 41$$

Поскольку время обработки изделий на втором станке согласно условию задачи не изменяется, то суммарное время обработки изделий можно уменьшить за счет изменения порядка запуска изделий в производство. Для решения этой задачи разработаны различные алгоритмы решения. Рассмотрим один из наиболее простых и часто используемых – алгоритм Джонсона.

Оптимизация запуска изделий в производство – это типичная комбинаторная задача, решение которой состоит в изменении порядка запуска изделий в производство. Следовательно, если перебрать все возможные варианты запуска изделий в производство (число вариантов N равно $N!$), то для нашей задачи $N = 6! = 720$.

6.4.3. Разработка последовательности действий и комплекса функций для расчета и изображения оптимальной загрузки двух станков

Алгоритм Джонсона значительно сокращает, а также упрощает процесс поиска оптимального варианта и состоит из трех этапов:

1. Поиска наименьших элементов. В табл. 6.2 находим минимальное время (это время обработки 2-го изделия на втором станке) и отмечаем его звездочкой.
2. Изменения порядка запуска изделий в производство.

Если наименьший элемент (время обработки изделия) относится к первому станку, то изделие первым запускается в производство и его перемещают в начало новой таблицы (см. табл. 6.3).

Если наименьший элемент (время обработки изделия) относится ко второму станку, то изделие запускается в производство последним и его перемещают в конец новой таблицы (см. табл. 6.3).

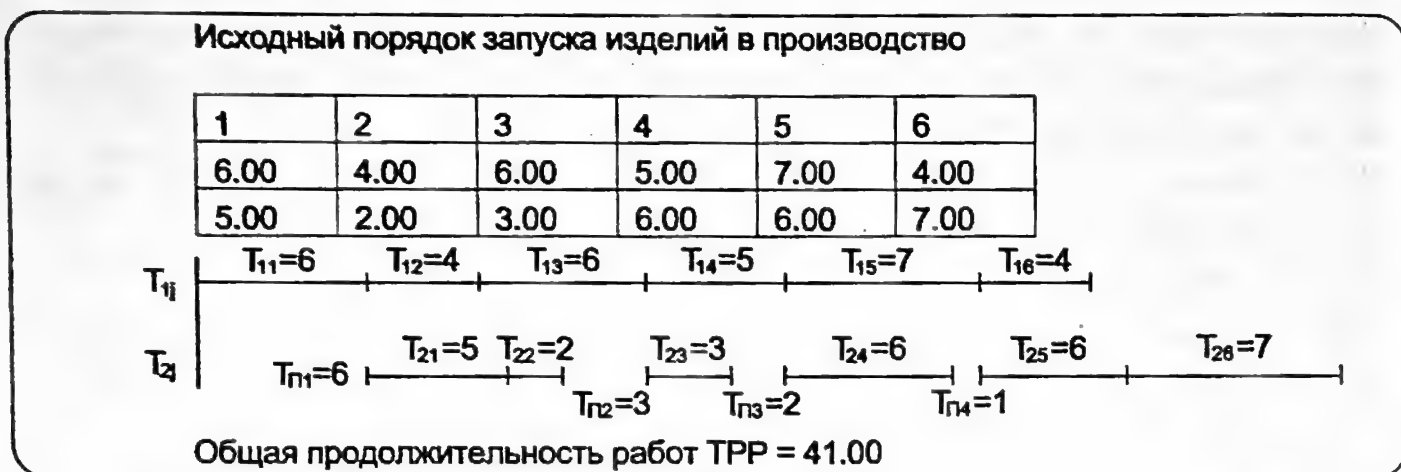


Рис. 6.26. График запуска изделий в производство

Когда имеются несколько наименьших элементов (время обработки изделия) и они относятся к первому (второму) станку, то изделие первым (последним) запускается в производство, то есть его перемещают в начало (конец) новой таблицы (табл. 6.3). Если наименьшие элементы (время обработки изделия) относятся как к первому станку, так и ко второму, то изделия перемещают в начало или конец новой таблицы согласно вышеизложенным правилам.

Перемещенное в начало (конец) таблицы изделие помечается звездочкой, переносу в дальнейшем не подлежит и в расчете не учитывается. В нашей задаче минимальный элемент относится ко второму станку, следовательно, изделие с минимальным временем обработки на втором станке обрабатывается последним (см. табл. 6.3). Это изделие, отмеченное звездочкой, как и весь столбец полностью, переносятся в конец табл. 6.3.

После перемещения изделия в начало (конец) таблицы алгоритм расчета повторяется с первого этапа до тех пор, пока изделия можно перемещать. Напомним, что перемещенные изделия (изделия, отмеченные звездочкой) в дальнейших расчетах не участвуют. В окончательном виде таблица запуска изделий в производство для нашей задачи представлена ниже (см. табл. 6.4).

Таким образом, оптимальный порядок запуска изделий в производство будет выглядеть следующим образом:

$$I_6 \rightarrow I_4 \rightarrow I_5 \rightarrow I_1 \rightarrow I_3 \rightarrow I_2$$

Таблица 6.3

$C_i \setminus I_j$	I_1	I_3	I_4	I_5	I_6	I_2^*
Станок 1	6	6	5	7	4	4
Станок 2	5	3	6	6	7	2

Таблица 6.4

$C_i \backslash I_j$	I_6	I_4^*	I_5	I_1^*	I_3^*	I_2^*
Станок 1	4	5	7	6	6	4
Станок 2	7	6	6	5	3	2

После определения оптимального порядка запуска изделий в производство вычисляется суммарное время изготовления всех изделий (см. рис. 6.27).

Суммарное время обработки изделий (Т) определяется как сумма времени простоев второго станка в ожидании изделий с первого станка и времени обработки этих изделий на втором станке.

$$T = \sum_{j=1}^M T_{2j} + \sum_{j=1}^M T_{n j} = (7 + 6 + 6 + 5 + 3 + 2) + (4 + 1) = 29 + 5 = 34$$

На рис. 6.27 видно, что время простоя второго станка существенно сократилось, следовательно сократилось время обработки всех изделий, оно составило Т = 34. Таким образом, суммарное время изготовления всех изделий сократилось на 17%.

Для расчета и создания графического изображения оптимального порядка запуска изделий в производство необходимо:

- определить местоположение заголовка, базовых точек (для создания графического изображения таблицы и графиков). Положим, что начало заголовка находится в точке ТТ с координатами Х = 10, а Y = 200. Базовая точка ввода данных в таблицу находится в точке ВТ с координатами Х = 10, а Y = 190. Начальными точками для прорисовки графиков являются ВТ1 и ВТ2, которые смещены по отношению к точке ВТ и ВТ1 вниз на 40 и 20 единиц соответственно;

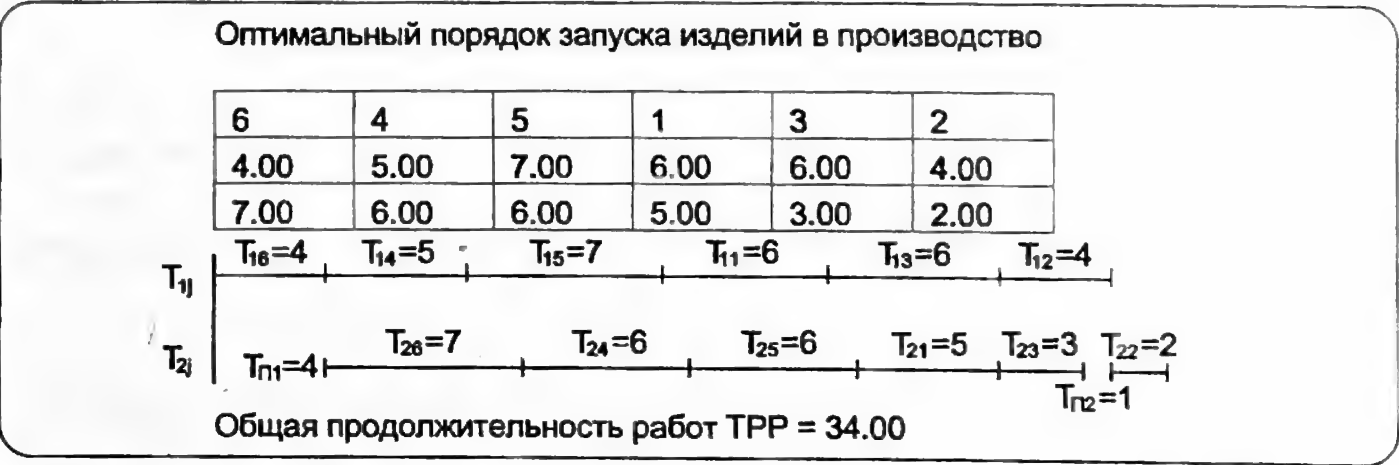


Рис. 6.27. График оптимального запуска изделий в производство

- *построить таблицы исходных данных и результатов расчета* для облегчения анализа данных. Построение будем осуществлять в цикле по числу изготавливаемых изделий с помощью специально созданной функции (**KLETKA**), которая строит клетку таблицы;
- *сформировать графическое изображение процесса изготовления изделий* на первом и втором станках для того, чтобы определить общую продолжительность всего процесса. График изготовления изделий может выходить за пределы экрана. Чтобы этого не случилось, следует определить коэффициент масштабирования КМ, с помощью которого корректируется время изготовления изделий;
- *изменить порядок изготовления изделий согласно алгоритму Джонсона* для оптимизации процесса;
- *отобразить график изготовления изделий на первом и втором станках* для того, чтобы определить общую продолжительность выполнения всего процесса.

Для отображения таблицы и графиков определим на AutoLISP местоположение заголовка и базовых точек:

```
(COMMAND "LIMITS" "0,0" "297,210" "ZOOM" "A")
(COMMAND "STYLE" "" "" "4" "1" "" "" "" "")
(SETQ TT '(10 200) ; базовая точка ввода заголовка таблицы
      BT '(10 190) ; базовая точка ввода данных в таблицу
      I 0)
(REPEAT 2 ; цикл изображения исходных
        ; данных и результатов
(SETVAR "BLIPMODE" 0) ; отключение изображения маркера
(IF (= I 0) (COMMAND "TEXT" TT "0"
  "Исходный порядок запуска изделий в производство")
  (COMMAND "TEXT" "10,100" "0"
  "Оптимальный порядок запуска изделий в производство")))
(SETQ BT1 (POLAR BT (* PI 1.5) 40) ; точка ввода графика
      BT2 (POLAR BT1 (* PI 1.5) 20) ; точка ввода графика
      BTT BT2
      TD 0)
```

Для ввода заголовков таблиц в функциях (**COMMAND...**) используется одна команда графического редактора **"TEXT"**.

Построение и ввод данных в таблицу можно организовать в виде цикла:

```
(FOREACH EL LT ; цикл построения таблицы с данными
  (SETQ N (NTH 0 EL) ; номер изготавливаемого изделия
        T1 (NTH 1 EL) ; время изготовления изделия на 1-м станке
        T2 (NTH 2 EL) ; время изготовления изделия на 2-м станке
        TD (+ TD T1 T2)) ; время для корректировки графика
```

```
(KLETKA BT) ; вызов функции для изображения клетки
(COMMAND "TEXT" (POLAR K4 (/ PI 9) 4) "0" (ITOA N))
(KLETKA K4)
(COMMAND "TEXT" (POLAR K4 (/ PI 9) 4) "0" (RTOS T1 2 2))
(KLETKA K4)
(COMMAND "TEXT" (POLAR K4 (/ PI 9) 4) "0" (RTOS T2 2 2))
(SETQ BT (POLAR BT 0 30)) ; базовая точка следующей клетки
)
```

Для создания клетки таблицы используется функция (KLETKA...), которая может быть записана так:

```
; Функция изображающая клетку таблицы
(DEFUN KLETKA (K1)
  (SETQ K2 (POLAR K1 0 30)
        K3 (POLAR K2 (* PI 1.5) 10)
        K4 (POLAR K1 (* PI 1.5) 10))
  (COMMAND "LINE" K1 K2 K3 K4 K1 ""))
)
```

Для вывода в таблицу числовых данных в функции (COMMAND...) используется команда графического редактора "TEXT".

Функция (ITOA N) преобразует целое число – номер изделия N в строковую константу, чтобы ввести затем в таблицу.

Функция (RTOS TMO 2 2) преобразует действительное число – время изготовления изделия в строковую константу, чтобы затем ввести это число в таблицу. Второй аргумент функции – 2 указывает, что число должно быть представлено в десятичном виде. Третий аргумент – 2 указывает, что число знаков после запятой должно быть равно двум.

Отображение графика изготовления изделий на первом и втором станках для исходного варианта и определение общей продолжительности всего процесса можно выполнить следующим образом:

```
(SETQ KM (/ 400 TD) ; определение коэффициента масштабирования
      T1 (NTH 1 (NTH 0 LT)) ; время изготовления 1-го изделия
      T1 (* T1 KM) ; скорректированное время
      BT2 (POLAR BT2 0 T1)) ; начало отображения 2-го графика
(SETVAR "BLIPMODE" 1) ; ввод маркера на графиках
(FOR EACH EL LT ; цикл отображения графиков
  (SETQ N (NTH 0 EL) ; номер изготавливаемого изделия
        T1 (NTH 1 EL) ; время изготовления изделия
              ; на 1-м станке
        T1 (* T1 KM) ; скорректированное время
              ; изготовления
        T2 (NTH 2 EL) ; время изготовления изделия
              ; на 2-м станке
```



```

T2 (* T2 KM) ; скорректированное время
; изготовления
BTK1 (POLAR BT1 0 T1)) ; точка окончания
; изготовления изделия
(COMMAND "PLINE" BT1 "W" 2 2 BTK1 "") ; отображение 1-го графика
(SETQ BT1 BTK1
BTT2 (POLAR BTK1 (* PI 1.5) 20))
(IF (>= (NTH 0 BTT2) (NTH 0 BT2)) (SETQ BT2 BTT2))
(SETQ BTK2 (POLAR BT2 0 T2))
(COMMAND "PLINE" BT2 "W" 2 2 BTK2 "") ; отображение 2-го графика
(SETQ BT2 BTK2)
)
(SETQ TPP (DISTANCE BTT BTK2)
TPP (/ TPP KM))
(COMMAND "TEXT" (POLAR BTT (* PI 1.5) 15) "0" (STRCAT
"Общая продолжительность работ TPP = " (RTOS TPP 2 2)))

```

Изменение порядка изготовления изделий согласно алгоритму Джонсона будет выглядеть следующим образом:

```

(SETQ BT '(10 90) ; точка ввода текста для
; оптимального варианта
L1 '() ; список с перемещением изделий
; в начало таблицы
L2 '() ; список с перемещением изделий
; в конец таблицы
NN (LENGTH LT)) ; определение числа изготавливаемых
; изделий

(REPEAT NN
(SETQ TMIN 1000.)
(FOREACH EL LT
(SETQ T1 (NTH 1 EL)
T2 (NTH 2 EL))
(IF (< T1 TMIN) (SETQ TMIN T1 NS 1))
(IF (< T2 TMIN) (SETQ TMIN T2 NS 2))
)
(FOREACH EL LT
(SETQ T1 (NTH 1 EL) ; время изготовления изделия
; на 1-м станке
T2 (NTH 2 EL)) ; время изготовления изделия
; на 2-м станке
(IF (AND (= T1 TMIN) (/= T1 T2) (= NS 1))
(SETQ L1 (CONS EL L1) LT (DEL EL LT)))
(IF (AND (= T2 TMIN) (= NS 2))
(SETQ L2 (CONS EL L2) LT (DEL EL LT))))

```

```
)
(SETQ LT (APPEND L1 L2)
  I (+ I 1))
)
```

Функция (CANS...) добавляет элемент EL в конец списка LT, а функция (DEL...) удаляет элемент EL из списка LT.

; Функция добавления элемента EL в конец списка LT

```
(DEFUN CANS (EL LT)
  (SETQ LT1 (REVERSE LT)
    LT2 (CONS EL LT1))
  (REVERSE LT2))
```

; Функция удаления одного элемента EL из списка LT

```
(DEFUN DEL (EL LT)
  (COND ((NULL LT) NIL)
    ((EQUAL EL (CAR LT)) (CDR LT))
    (T (CONS (CAR LT) (DEL EL (CDR LT))))))
```

В окончательном виде функция (DJON...), которая обеспечивает расчет и графическое отображение оптимальной загрузки двух станков (алгоритм Джонсона), представлена ниже.

6.4.4. Функция для расчета и изображения процесса оптимальной загрузки двух станков

```
; *****
```

; Функция (DJON LT) обеспечивает расчет и графическое отображение

; оптимальной загрузки двух станков (алгоритм Джонсона)

; А р г у м е н т ы ф у н к ц и и:

; LT - список норм времени выполнения каждой операции на

; двух станках ((1 T11 T21) (2 T12 T22) (3 T13 T23)...))

; TIJ... - время изготовления J-го изделия на I-м станке

; Пример использования функции (DJON LT)

; (DJON '((1 6 5) (2 4 2) (3 6 3) (4 5 6) (5 7 8) (6 4 7)))

```
; *****
```

```
(DEFUN DJON (LT)
```

```
  (SETVAR "CMDECHO" 0) ; отключение эха команд
```

```
  (COMMAND "LIMITS" "0,0" "297,210")
```

```
  (COMMAND "ERASE" "ALL" ""))
```

```
  (COMMAND "STYLE" "" "" "4" "1" "" "" "" ""))
```

```
  (SETQ TT '(10 200) ; базовая точка ввода заголовка таблицы
```

```
    BT '(10 190) ; базовая точка ввода данных в таблицу
```

```

I 0)
(REPEAT 2 ; цикл отображения исходных данных
; и результатов
(SETVAR "BLIPMODE" 0) ; отключение отображения маркера
(IF (= I 0) (COMMAND "TEXT" TT "0"
"Исходный порядок запуска изделий в производство")
(COMMAND "TEXT" "10,100" "0"
"Оптимальный порядок запуска изделий в производство"))
(SETQ BT1 (POLAR BT (* PI 1.5) 40) ; точка ввода графика
BT2 (POLAR BT1 (* PI 1.5) 20) ; точка ввода графика
BTT BT2
TD 0)
(FOR EACH EL LT ; цикл построения таблицы с данными
(SETQ N (NTH 0 EL) ; номер изготавливаемого изделия
T1 (NTH 1 EL) ; время изготовления изделия на 1-м станке
T2 (NTH 2 EL) ; время изготовления изделия на 2-м станке
TD (+ TD T1 T2)) ; время для корректировки графика
(KLETKA BT) ; вызов функции для изображения клетки
(COMMAND "TEXT" (POLAR K4 (/ PI 9) 4) "0" (ITOA N))
(KLETKA K4)
(COMMAND "TEXT" (POLAR K4 (/ PI 9) 4) "0" (RTOS T1 2 2))
(KLETKA K4)
(COMMAND "TEXT" (POLAR K4 (/ PI 9) 4) "0" (RTOS T2 2 2))
(SETQ BT (POLAR BT 0 30)) ; базовая точка следующей клетки
)
(SETQ KM (/ 400 TD) ; определение коэффициента
; масштабирования
T1 (NTH 1 (NTH 0 LT)) ; время изготовления 1-го изделия
T1 (* T1 KM) ; скорректированное время
BT2 (POLAR BT2 0 T1)) ; начало изображения 2-го графика
(SETVAR "BLIPMODE" 1) ; ввод маркера на графиках
(FOR EACH EL LT ; цикл отображения графиков
(SETQ N (NTH 0 EL) ; номер изготавливаемого изделия
T1 (NTH 1 EL) ; время изготовления изделия
; на 1-м станке
T1 (* T1 KM) ; скорректированное время изготовления
T2 (NTH 2 EL) ; время изготовления изделия
; на 2-м станке
T2 (* T2 KM) ; скорректированное время изготовления
BTK1 (POLAR BT1 0 T1)) ; точка окончания
; изготовления изделия
(COMMAND "PLINE" BT1 "W" 2 2 BTK1 "") ; 1-й график
(SETQ BT1 BTK1

```

```

    BTT2 (POLAR BTK1 (* PI 1.5) 20))
(IF (>= (NTH 0 BTT2) (NTH 0 BT2)) (SETQ BT2 BTT2))
  (SETQ BTK2 (POLAR BT2 0 T2))
  (COMMAND "PLINE" BT2 "W" 2 2 BTK2 "") ; 2-й график
  (SETQ BT2 BTK2)
)
(SETQ TPP (DISTANCE BTT BTK2)
  TPP (/ TPP KM))
(COMMAND "TEXT" (POLAR BTT (* PI 1.5) 15) "0" (STRCAT
  "Общая продолжительность работ TPP = " (RTOS TPP 2 2)))
(SETQ BT '(10 90) ; точка ввода текста для
  ; оптимального варианта

  L1 '()
  L2 '()
  NN (LENGTH LT))
(REPEAT NN
  (SETQ TMIN 1000.)
  (FOREACH EL LT
    (SETQ T1 (NTH 1 EL)
      T2 (NTH 2 EL))
    (IF (< T1 TMIN) (SETQ TMIN T1 NS 1))
    (IF (< T2 TMIN) (SETQ TMIN T2 NS 2))
  )
  (FOREACH EL LT
    (SETQ T1 (NTH 1 EL)
      T2 (NTH 2 EL))
    (IF (AND (= T1 TMIN) (/= T1 T2) (= NS 1))
      (SETQ L1 (CONS EL L1) LT (DEL EL LT)))
    (IF (AND (= T2 TMIN) (= NS 2))
      (SETQ L2 (CONS EL L2) LT (DEL EL LT))))
  )
(SETQ LT (APPEND L1 L2) ; новый порядок запуска изделий
  I (+ I 1)) ; в производство
)
(COMMAND "ZOOM" "ALL" "")
)
; Функция изображения клетки таблицы
(DEFUN KLETKA (K1)
  (SETQ K2 (POLAR K1 0 30)
    K3 (POLAR K2 (* PI 1.5) 10)
    K4 (POLAR K1 (* PI 1.5) 10))
  (COMMAND "LINE" K1 K2 K3 K4 K1 "")
)

```

; функция добавления элемента EL в конец списка LT

```
(DEFUN CANS (EL LT)
  (SETQ LT1 (REVERSE LT)
        LT2 (CONS EL LT1))
  (REVERSE LT2)
)
```

; функция удаления одного элемента EL из списка LT

```
(DEFUN DEL (EL LT)
  (COND ((NULL LT) NIL)
        ((EQUAL EL (CAR LT)) (CDR LT))
        (T (CONS (CAR LT) (DEL EL (CDR LT))))))
)
```

(DJON '((1 6 5) (2 4 2) (3 6 3) (4 5 6) (5 7 6) (6 4 7))) ; вызов функции
Результат выполнения функции (DJON...) представлен на рис. 6.28.

Исходный порядок запуска изделий в производство

1	2	3	4	5	6
6.00	4.00	6.00	5.00	7.00	4.00
5.00	2.00	3.00	6.00	6.00	7.00

Общая продолжительность работ TPP = 41.00

Оптимальный порядок запуска изделий в производство

6	4	5	1	3	2
4.00	5.00	7.00	6.00	6.00	4.00
7.00	6.00	6.00	5.00	3.00	2.00

Общая продолжительность работ TPP = 34.00

Рис. 6.28. Оптимизация запуска изделий в производство

ОТЛАДКА И СОЗДАНИЕ ПРИЛОЖЕНИЙ В СРЕДЕ VISUAL LISP

Отладка программ в среде Visual LISP	300
Создание приложений в среде Visual LISP	321
Встроенные функции Visual LISP	332

В этой главе описывается отладчик Visual LISP, приемы создания автономных приложений в среде Visual LISP. Здесь же рассматриваются встроенные функции интегрированной среды Visual LISP.

7.1. Отладка программ в среде Visual LISP

Отладка – как правило, наиболее трудоемкая стадия в разработке любой программы. Visual LISP включает мощный отладчик, который позволяет:

- запустить программу **Trace** (трассировка программы);
- отслеживать значения переменных во время выполнения программы;
- просмотреть последовательность значений выполняемых выражений (функций);
- просмотреть значения параметров, используемых внутри функциональных обращений;
- запустить программу прерываний **Interrupt**;
- производить пошаговое выполнение программы;
- просмотреть содержимое стека.

Для решения этих задач Visual LISP имеет следующие средства:

- **Break loop mode** (Режим прерываний в цикле), который позволяет останавливать выполнение программы в определенных точках, просматривать и изменять значения объектов (переменных, символов, функций и выражений) во время прерывания;
- **Inspectors** (Инспектора), которые позволяют получать детализированную информацию об объектах в окне **Inspector dialog Window** (Диалоговое окно просмотра);
- **Watch Window** (Окно наблюдений), которое позволяет наблюдать значения переменных во время выполнения программы. Содержание окна наблюдений Watch Window модифицируется автоматически;
- **Trace Stack Facility** (Средство трассировки стека), которое позволяет просматривать функциональный стек вызовов – механизм, с помощью которого Visual LISP записывает последовательность выполнения функций;
- **Trace Facility** (Средство трассировки), которое позволяет регистрировать обращения и значения отслеживаемых функций в специальном окне **Trace** (Трассировка).

Отладчик может приостановить выполнение программы до или после выполнения любого выражения.

Управлять отладкой можно из разных мест Visual LISP, включая текстовый редактор, окно консоли, различные меню и кнопки инструментальной панели **Debug** (рис. 7.1).



Рис. 7.1. Кнопки инструментальной панели Debug

Щелкните по кнопке **Load active edit window** (Загрузить текст активного окна редактирования) на инструментальной панели **Tools** (Инструменты) или выберите пункт **Load Text in Editor** (Загрузить текст из редактора) падающего меню **Tools** (Инструменты). На экране появится окно консоли Visual LISP (рис. 7.5).

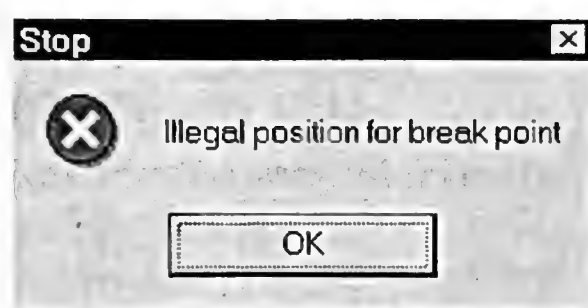


Рис. 7.4. Окно Stop

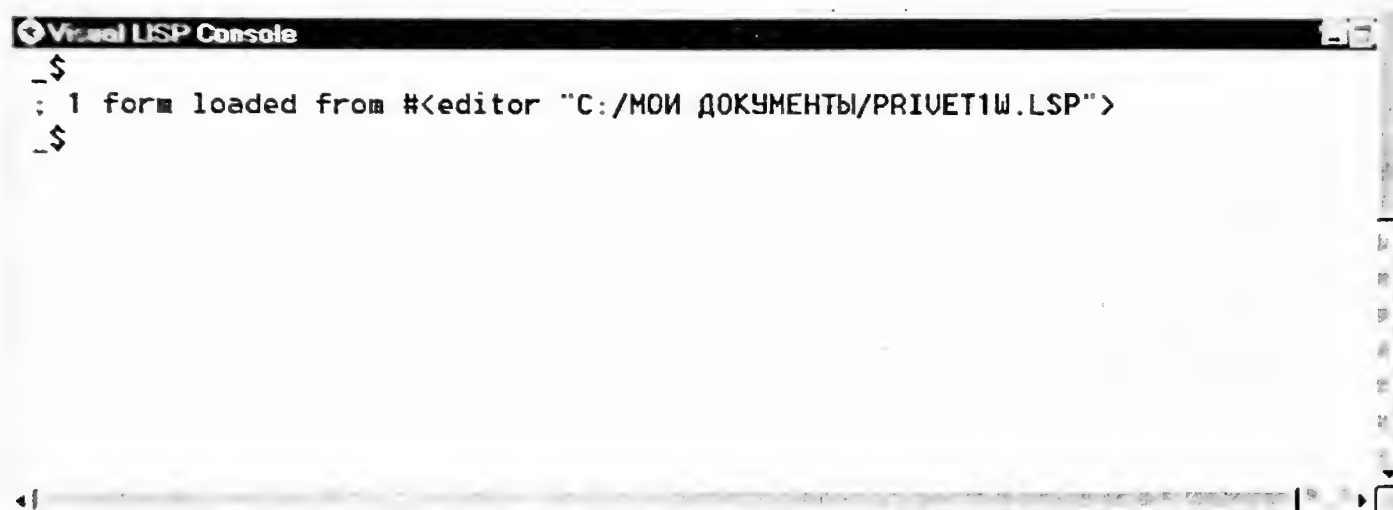


Рис. 7.5. Окно консоли Visual LISP

Загрузите и выполните в окне консоли Visual LISP функцию (**PRIVET**).

Во время ее выполнения отладчик возвращает пользователя в окно текстового редактора и останавливает выполнение функции (**PRIVET...**) в точке прерывания, которую пользователь установил ранее, выделив все выражение (рис. 7.6).

Можно последовательно продвигаться по программе, выполняя одно или несколько выражений до установленной точки прерывания. Для этого щелкните по кнопке **Step Into** (Шаг внутрь) или выберите одноименный

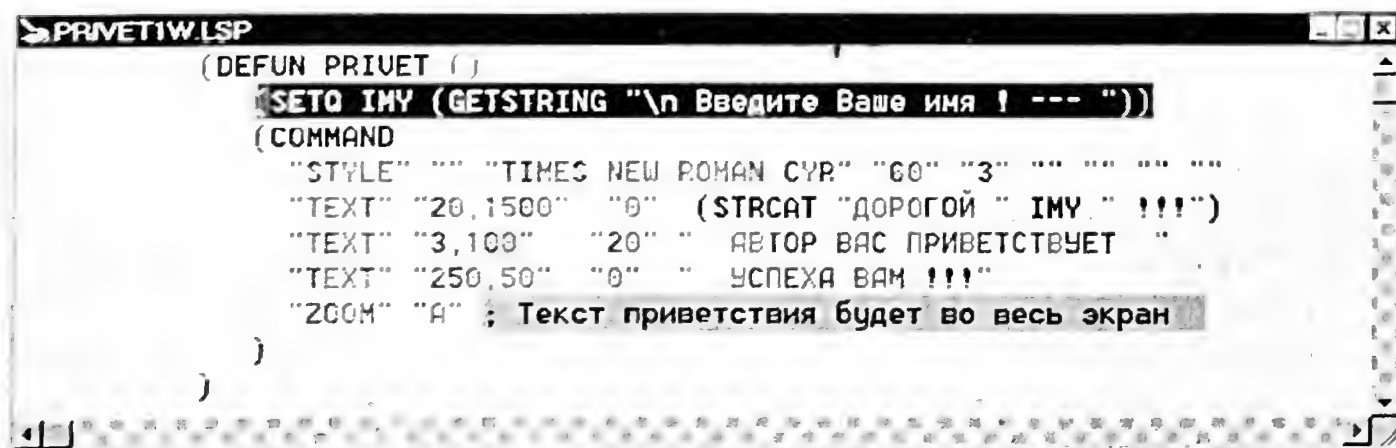


Рис. 7.6. Окно текстового редактора

Можно сделать шаг с выполнением всего выделенного выражения. Для этого следует щелкнуть по кнопке **Step Over** (Шаг через) или выбрать одноименный пункт из падающего меню **Debug** (Отладка) (**Alt+D**). Можно также нажать комбинацию клавиш **Shift+F8**. Кнопка **Step Over** позволяет оценить выражение со всеми вложенными выражениями, а затем остановиться в конце всего выражения. При этом курсор перемещается в конец выражения.

Результат этих действий можно увидеть, если активизировать систему AutoCAD путем нажатия кнопки **Activate AutoCAD** (Активизировать AutoCAD) или выбора одноименной команды падающего меню **Window** (**Alt+W**).

При необходимости можно отследить также значения каждого выражения. Чтобы проверить значение последнего выполненного выражения, выберите пункт **Watch Last Evaluation** (Наблюдение последнего значения) из падающего меню **Debug** (Отладка). На экране появится окно **Watch** (Окно наблюдений), содержащее значение системной переменной ***LAST-VALUE***, в которой Visual LISP всегда сохраняет значение последнего выполненного выражения (рис. 7.9).

Чтобы отслеживать значения переменных во время выполнения программы, можно использовать окно **Watch** (Окно наблюдений). Для этого в окне текстового редактора, где открыт, например, файл **PRIVET1W.LSP**, дважды щелкните по любому текущему имени переменной, к примеру, **IMY**.

Если щелкнуть по кнопке **Add Watch** (Добавить наблюдение), Visual LISP передаст имя переменной **IMY** окну **Watch** и отобразит текущее значение переменной в окне.

Допустим, окно **Watch** (Окно наблюдений) не было открыто, однако необходимо просмотреть значения переменной. В таком случае окно можно открыть, если выбрать пункт **Watch Window** (Окно наблюдений) из падающего меню **View** (Просмотр) Visual LISP.

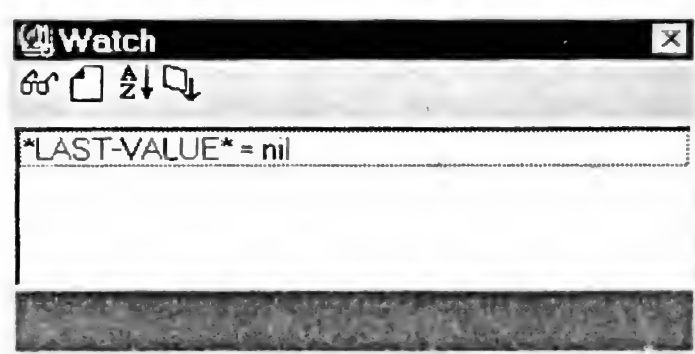


Рис. 7.9. Окно значения переменной ***LAST-Value***

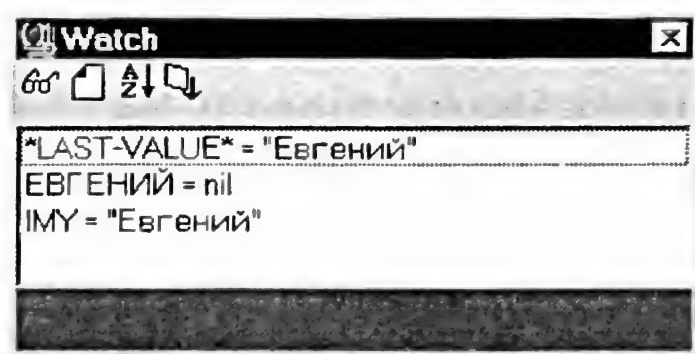


Рис. 7.10. Окно наблюдений значений переменных

Установить наблюдение за переменной можно и другим способом. Щелкните по кнопке **Add Watch** (Добавить наблюдение) – это первая кнопка (с очками) в окне Watch. На экране появится окно **Enter expression to watch** (Введите выражение для наблюдений), как показано на рис. 7.11.

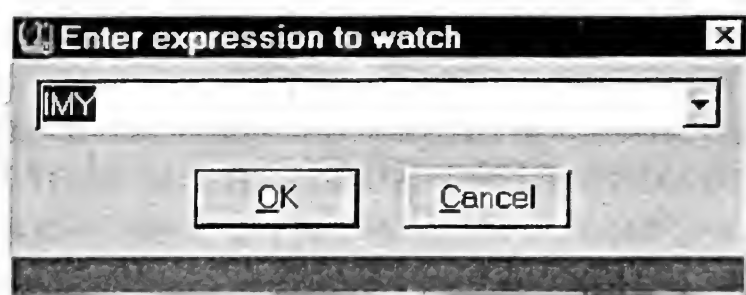


Рис. 7.11. Окно ввода выражения для наблюдений

В этом окне можно ввести имя переменной, которую необходимо просмотреть. Самое близкое к курсору имя переменной Visual LISP копирует в окно автоматически. Если вы хотите просматривать другую переменную, введите ее имя. Visual LISP модифицирует значения переменных в окне **Watch** (Окно наблюдений) после выполнения каждого шага.

Инициализировать шаг выполнения можно, если щелкнуть мышкой по кнопке **Step Over** (Шаг через) или дважды нажать комбинацию клавиш **Shift+F8**. Значение IMY в окне Watch (Окно наблюдений) изменится (до выполнения оно было NIL).

Чтобы продолжить выполнение программы до следующей точки прерывания или до конца (если точек прерывания нет), щелкните по кнопке **Continue** (Продолжить) на инструментальной панели **Debug** (Отладка) или выберите пункт **Continue** (Продолжить) падающего меню **Debug** меню Visual LISP.

Кроме установки точек прерывания Visual LISP обеспечивает следующие режимы управления выполнением программы:

- **Break On Error** (Прерывание по ошибке) – активизирует автоматическое прерывание всякий раз, когда во время выполнения программы возникает ошибка. Для включения этого режима необходимо выбрать пункт **Break On Error** (Прерывание по ошибке) падающего меню **Debug** меню Visual LISP;
- **Stop Once** (Остановка сразу) – заставляет Visual LISP остановиться, когда выполняется самое первое выражение LISP. Для включения этого режима необходимо выбрать пункт **Stop Once** (Остановиться сразу) падающего меню **Debug**;
- **Break on function entry** (Прерывание на входе функции). Во время прерывания текст функции будет отображаться в специальном окне. При этом можно включить или выключить флажок **Debug-On-Entry** (Отладка на входе) в интерактивном диалоговом окне **Symbol Service** (Обслуживание символов) или поместить внутри программы псевдокомментарии;

- **Debug top-level mode** (Режим отладки верхнего уровня) – позволяет управлять загрузкой программы из файла или из окна редактора. Прерывания происходят перед выполнением каждого выражения верхнего уровня (типа DEFUN). Для включения этого режима необходимо выбрать пункт **General Options** (Общие параметры) всплывающего меню пункта **Environment Options** (Параметры среды) падающего меню **Tools** (Инструменты) главного меню Visual LISP. Затем щелкнуть по закладке **Diagnostic** (Диагностика) и убрать флажок в переключателе **Do not debug top-level** (Не отлаживать на верхнем уровне).

Обратите внимание, если режимы **Debug top-level mode** (Режим отладки верхнего уровня) и **Stop Once** (Остановка сразу) включены, Visual LISP переходит в режим отладки каждый раз, когда загружается файл;

- **Animate** (Оживить) – предполагает неоднократное использование команды **Step Into** (Шаг внутрь) с определенной задержкой. Окно редактора в режиме **Animate** отображает выполняемые выражения, а окно **Watch** (Окно наблюдений) постоянно модифицирует значения переменных. Чтобы включить режим **Animate**, выберите пункт **Animate** (Оживить) падающего меню **Debug** (Отладка) главного меню Visual LISP. Когда режим **Animate** включен, режим **Stop Once** (Остановить сразу) игнорируется.

Быстродействие анимации управляется значением задержки анимации. Для просмотра и установки задержки выберите пункт **General Options** (Общие параметры) из всплывающего меню пункта **Environment Options** (Параметры среды) падающего меню **Tools** (Инструменты) главного меню Visual LISP и щелкните по закладке **Diagnostic** (Диагностика). Время задержки можно изменять в окошке **Animate delay** (Задержка анимации).

Самый простой способ начать отладку – выбрать пункт **Stop Once** (Остановить сразу) падающего меню **Debug** (Отладка) главного меню Visual LISP. Выполнение первого выражения LISP будет прервано. Продолжить выполнение программы можно с помощью команд отладчика. Кроме того, начать отладку можно путем установки точек прерывания, как было показано ранее. Во время прерывания соответствующее окно редактора текста Visual LISP показывает текущее выражение LISP в точке прерывания. Маркер прерывания появится в окне консоли, используя которое, можно изменять и модифицировать программу, где произошло прерывание. В окне **Watch** (Окно наблюдений) можно также исследовать переменные.

Когда программа AutoLISP выполняется без включения отладки, Visual LISP выполняет ее в режиме **Top Level** (Верхний уровень) и **Read-Eval-Print** (Прочитать-Выполнить-Напечатать). В ходе выполнения выражения в окне консоли Visual LISP отображается подсказка.

В случае если выполнение программы прерывается или приостанавливается, в окне консоли Visual LISP включается система управления проходами. На экране появляется подсказка, которая начинается с числа, указывающего уровень прерывания. Например, когда прерывание возникает впервые, перед подсказкой появляется `_1`:

`_1_$`


При выходе из состояния прерывания (например, после выдачи команды **"QUIT"** (Выйти)), текущий **Read-Eval-Print** (Прочитать-Выполнить-Напечатать) цикл завершается, а предыдущий продолжается. Если значение переменной изменяется в текущем цикле прерывания, оно будет использовано, когда выполнение продолжится.


В Visual LISP существуют прерывания с продолжением (**Continuable**) и без продолжения (**Noncontinuable**). Продолжать выполнение можно после следующих видов прерывания:


- режима **Stop Once** (Остановка сразу);
- достижения функции, отмеченной для **Debug on entry** (Отладка на входе);
- достижения точки прерывания, установленной в программе;
- щелчка по кнопке **Pause**;
- режимов **Step Over** (Шаг через), **Step Into** (Шаг внутрь) или **Step Out** (Шаг вне).

Во время прерывания при активном окне консоли и при отражении в подсказке текущего уровня прерывания имеется доступ для чтения-записи ко всем переменным среды, в которой произошло прерывание. Когда, например, прерывание произошло внутри функции, содержащей несколько локальных переменных, изменить их значения можно, если использовать в строке подсказок в окне консоли функцию присваивания (**SETQ...**).




Управлять последующим выполнением программы при остановке выполнения программы в точке прерывания можно, если выбрать одну из следующих команд падающего меню **Debug** (Отладка) или щелкнуть эквивалентную кнопку на панели инструментов:

 **Reset to the top level** (Возвратить на верхний уровень) – завершить прерывание и возвратиться на самый верхний уровень **Read-Eval-Print** (Прочитать-Выполнить-Напечатать);

 **Quit current level** (Выйти из текущего уровня) – завершить прерывание и возвратиться на один уровень вверх (уровень прерывания либо уровень **Read-Eval-Print** (Прочитать-Выполнить-Напечатать));

 **Continue** (Продолжить) – продолжить выполнение программы с точки прерывания.

Пункты меню, которые начинаются со **Step** (Шаг), позволяют управлять пошаговым выполнением выражения:

-  **Step Over** (Шаг через) – с выполнением всех вложенных выражений;
-  **Step Into** (Шаг внутрь) – с заходом в первое вложенное выражение;
-  **Step Out** (Шаг вне) – с переходом во внешнее выражение. При этом выполняются все остающиеся вложенные выражения.

После выхода из цикла прерывания к верхнему уровню консоли подсказка консоли возвращается к ее первоначальному виду (без числового префикса).

Точка прерывания позволяет отмечать позицию в программе, в которой выполнение программы должно быть прервано. Устанавливать прерывания можно до или после анализируемых выражений.

Точки прерывания могут быть установлены только из окна текстового редактора Visual LISP. Чтобы установить или удалить точку прерывания, переместите курсор в позицию, в которой следует остановить выполнение программы. Если необходимо остановить выполнение перед открывающей скобкой выражения, поместите курсор слева от выражения, щелкните кнопку **Toggle breakpoint** (Установить/удалить точку прерывания) на инструментальной панели **Debug** (Отладка) или нажмите клавишу **F9**, чтобы установить точку прерывания. Для установки точки прерывания выберите пункт **Toggle breakpoint** (Установить/удалить точку прерывания) из всплывающего меню **Debug** (Отладка) главного меню Visual LISP или щелкните правой кнопкой мышки и выберите пункт **Toggle breakpoint** (Установить/удалить точку прерывания) из всплывающего контекстного меню. **Toggle breakpoint** работает как переключатель «вкл\выкл».

Если курсор помещается в неопределенную позицию, например, в середину выражения, Visual LISP переместит курсор к самой близкой круглой скобке, и на экране появится запрос **Set break point here?** (Установить точку прерывания здесь?).

Когда вы согласны с местом установки точки прерывания, введите **Yes** (Да) либо **No** (Нет), если хотите установить точку прерывания сами.

Позицию каждой точки прерывания Visual LISP отмечает цветным прямоугольником, так что различать точки прерывания в программе несложно. По умолчанию активные точки прерывания отмечены красным цветом, который можно изменить. Для этого выберите пункт **Configure Current...** (Текущая конфигурация) всплывающего меню пункта **Window Attributes** (Параметры окна) падающего меню **Tools** (Инструменты). На экране появится диалоговое окно **Window Attributes** (Параметры окна),

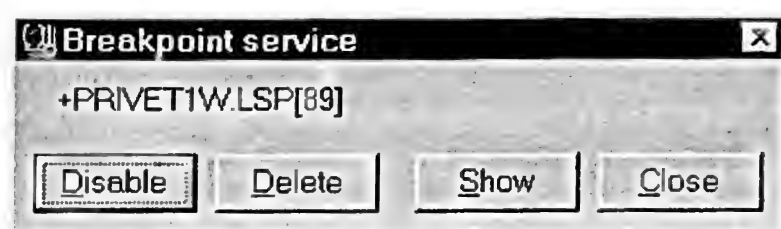


Рис. 7.12. Окно обслуживания точки прерывания Breakpoint service

где в раскрывающемся списке следует выбрать пункт (строку) **BPT-ACTIVE**, а затем желаемый цвет.

При использовании нескольких точек прерывания внутри файла полезно отключить на время одну или большее количество точек прерывания. Однако для использования точки прерывания в дальнейшем ее позицию необходимо оставить. Это сокращает время на удаление и восстановление точек прерывания.

Чтобы отключить точку прерывания, установите курсор на маркер точки прерывания и щелкните правой кнопкой мышки. Из всплывающего меню выберите пункт **Breakpoint service** (Обслужить точку прерывания). Visual LISP покажет диалоговое окно обслуживания точки прерывания **Breakpoint service** (Обслужить точку прерывания), показанное на рис. 7.12.

Чтобы временно отключить точку прерывания, щелкните кнопку **Disable** (Отключить) в окне **Breakpoint service** (Обслужить точку прерывания). При отключении Visual LISP изменяет цвет маркера точки прерывания. По умолчанию он отмечает заблокированные точки прерывания синим цветом, который также можно изменить. Для этого выберите пункт **Configure Current...** (Текущая конфигурация) всплывающего меню пункта **Window Attributes** (Параметры окна) падающего меню **Tools** (Инструменты). На экране появится диалоговое окно **Window Attributes** (Параметры окна). В этом окне в раскрывающемся списке выберите пункт (строку) **BPT-DISABLE**, а затем желаемый цвет.

Пункт **Breakpoints Window** (Окно точек прерывания) падающего меню **View** (Просмотр) позволяет увидеть список всех точек прерывания, в настоящее время определенных Visual LISP (рис. 7.13).

Окно **Breakpoints** содержит точки прерывания всех файлов, которые редактируются в Visual LISP, а не только файла из активного окна редактора. В примере на рисунке 7.13 указан один файл, потому что только он в настоящее время открыт. Установку точек прерывания можно изменять в любом количестве открытых файлов.

Каждая строка в окне **Breakpoints** содержит имя файла, имеющего точку прерывания, и указывает расположение этой точки. Знак + или – определяет, активна указанная точка прерывания или заблокирована.

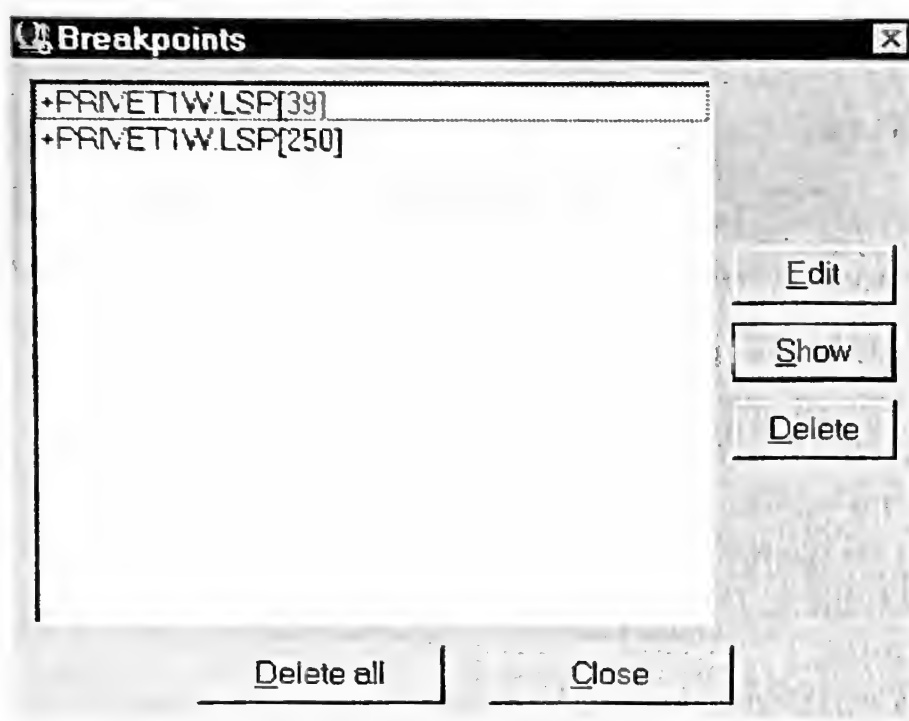


Рис. 7.13. Окно списка точек прерывания

Диалоговое окно позволяет удалять все точки прерывания сразу, редактировать или отображать одну из них. С помощью кнопки **Show** (Показать) можно отображать позицию точки прерывания. Кнопка **Edit** (Редактирование) открывает диалоговое окно обслуживания точки прерывания **Breakpoint service** (Обслуживание точки прерывания) после выделения редактируемой строки (см. рис. 7.12).

Устанавливать точки прерывания можно до либо после загрузки программы. Однако, если вы изменяете текст в программе после ее загрузки, а затем добавляете точку прерывания, то данная точка будет действовать только после перезагрузки программы.

Установки точек прерывания остаются в силе в течение всего сеанса редактирования Visual LISP. Сохранить установки для других сеансов можно с помощью команды **Save Settings** (Сохранить установки) падающего меню **Tools** (Инструменты).

Точки прерывания теряются автоматически в том случае, если выполняются следующие действия:

- удаление фрагмента программы, содержащего точку прерывания;
- изменение файла не в среде редактора Visual LISP (например, с помощью записной книжки);
- применение форматирования Visual LISP к фрагментам программ, содержащим точки прерывания.

Visual LISP обеспечивает почти безграничный доступ к символам и их значениям из любого места программы с помощью следующих окон:

- **Watch** (Окно наблюдений), которое отображает текущее значение любого набора переменных;
- **Trace Stack** (Трассировка стека), которое отображает самую современную иерархию обращения. На любом уровне стека вы можете просматривать соответствующий код, код вызова, локальные переменные и другие элементы;
- **Symbol Service** (Обслуживание символов), которое отображает текущее значение символа так же, как и его текущих флажков. Вы можете изменять значение и/или флажки;
- **Inspect:** (Проверка), которое проверяет любой элемент выражения AutoLISP;
- **Frame binding window** (Рамка, связывающая окна), которая связывает окна и отображает значения всех локальных переменных.

Чтобы получить возможность просматривать значения переменных, необходимо добавить переменную в окно **Watch** (Окно наблюдений). Когда курсор находится на имени переменной, двойным щелчком мышки выделите ее в любом контексте (в окне редактора, консоли и т. д.) и щелкните по кнопке **Add Watch** (Добавить наблюдение) или выберите пункт **Add Watch** (Добавить наблюдение) падающего меню **Debug** (Отладка). Можно также выбрать пункт **Add Watch** из контекстного (всплывающего) меню. Для этого нажмите правую кнопку мышки, когда курсор находится на имени переменной. На экране появится диалоговое окно **Enter expression to watch** (Введите выражение для наблюдений) с введенным именем переменной. Щелкните по кнопке **OK** (Okay – хорошо).

Если окно **Watch** (Окно наблюдений) уже активно, вы можете добавить дополнительные переменные к списку **Watch**, щелкая по кнопке **Add Watch** (Добавить наблюдение) – кнопке с очками на инструментальной панели в окне **Watch** (Окно наблюдений).

Когда Visual LISP не сможет определить, какая переменная вас заинтересовала, на экране появится диалоговое окно **Enter expression to w...** (Ввод выражения для наблюдений). Введите в текстовом поле раскрывающегося списка (**Drop-down list**) нужное имя переменной и щелкните по кнопке **OK**.

Инструментальная панель окна **Watch** (Окно наблюдений) содержит следующие кнопки:

- **Add Watch** (Добавить наблюдение) – вызывает пункт **Add Watch**, чтобы добавить новую переменную в окно **Watch**; эта переменная может быть выбрана в любом активном текстовом окне или напечатана в диалоговом окне **Add Watch** (Добавить наблюдение);

- **Clear Window** (Очистить окно) – очищает окно Watch (Окно наблюдения);
- **Sort expressions** (Отсортировать выражения) – сортирует переменные в окне Watch (Окно наблюдений) в алфавитном порядке по имени;
- **Copy to Trace/Log** (Копировать в окно Trace или Log) – позволяет сохранять содержимое диалогового окна Watch (Окно наблюдений) в окне Trace (Трассировка) или файле Log (Файл регистрации).

Для работы с переменными и их значениями в окне Watch можно использовать контекстное меню, которое вызывается щелчком правой кнопки мышки (рис. 7.14).

Контекстное меню окна наблюдений Watch содержит следующие пункты:

- **Inspect value** (Проверить значение) – просмотреть значение;
- **Copy value** (Скопировать значение) – скопировать значение выбранной переменной в переменную системы *OBJ*;
- **Print value** (Напечатать значение) – напечатать значение переменной в окне консоли с одиночной кавычкой (') впереди;
- **Symbol...** (Символ) – вызвать диалоговое окно обслуживания символов для выбранной переменной;
- **Apropos...** (Поиск по фрагменту) – используя имя выбранного символа или нескольких символов, вызвать диалоговое окно Apropos... как параметр поиска полного имени;
- **Remove from Watch** (Удалить из окна наблюдений) – удалить выбранную переменную из окна наблюдений Watch.

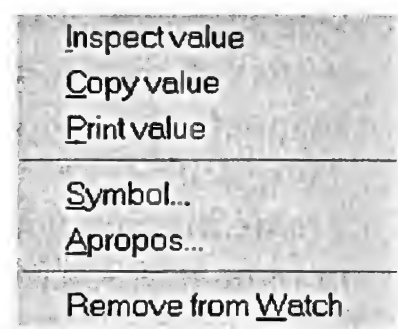


Рис. 7.14. Контекстное меню окна Watch

Visual LISP имеет специальное средство отладки **Trace Stack** (Трассировка стека). Трассировка стека – история выполнения функций внутри программы. Структура стека часто упоминается как **LIFO – Last In, First Out** (Последний вошел, первый вышел).

Трассу стека полезно исследовать при двух следующих условиях:

- когда программа находится в состоянии ожидания, например, во время контрольной паузы;
- после того, как произошла ошибка.

Допустим, в файле PRIVET1W.LSP во второй строке пропущена вторая закрывающая скобка. После загрузки и запуска на исполнение в окне консоли появится сообщение: **malformed list on input** (плохо сформированный список на вводе), как на рис. 7.15. Это означает, что имеющихся закрывающих круглых скобок недостаточно.

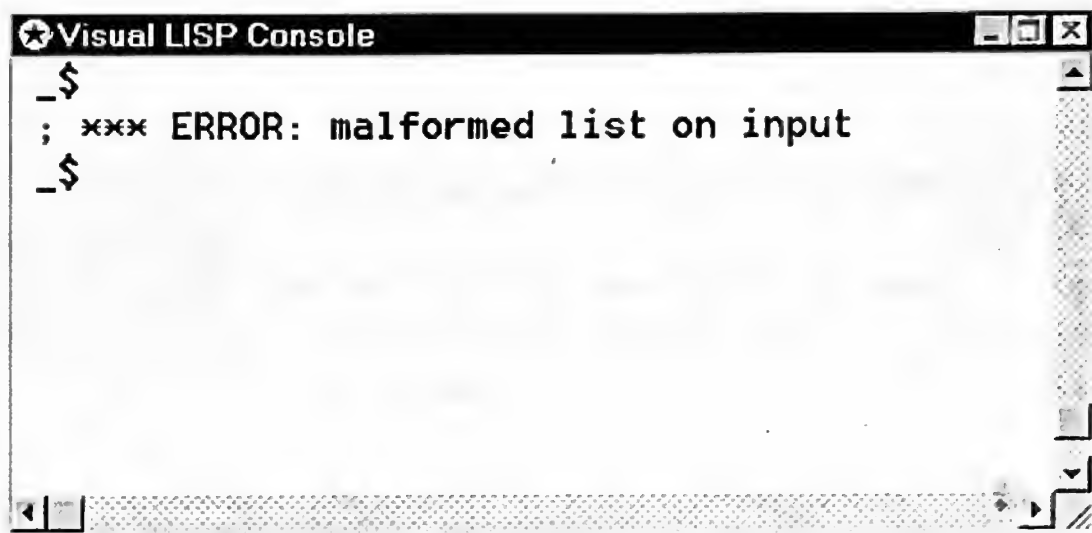


Рис. 7.15. Окно консоли

Чтобы увидеть состояние функциональных вызовов до того, как ваша программа завершится из-за ошибки, выберите пункт **Error trace** (Трассировка ошибки) падающего меню **View** (Просмотр). На экране появится диалоговое окно **Error trace** (Трассировка ошибки), как показано на рис. 7.16.

Окно **Error trace** используется Visual LISP, чтобы запомнить выход из вложенного ряда выражений. Изучая трассировку, можно видеть, что происходило внутри программы в процессе ее выполнения, во время или сразу после того, как она завершилась по ошибке.

Чтобы наблюдать состояние функционального стека вызовов во время прерывания, выберите пункт **Trace Stack** (Трассировка стека) падающего меню **View** главного меню Visual LISP или щелкните по кнопке **Trace** (Трассировка) – кнопке с отпечатками следов ног. На экране появится диалоговое окно **Trace Stack** (Трассировка стека), как показано на рис. 7.17.

Окно **Trace Stack** включает функцию (она называется фреймом) для отслеживания стека функции. Второй элемент, или фрейм в стеке следа, подсвечен.

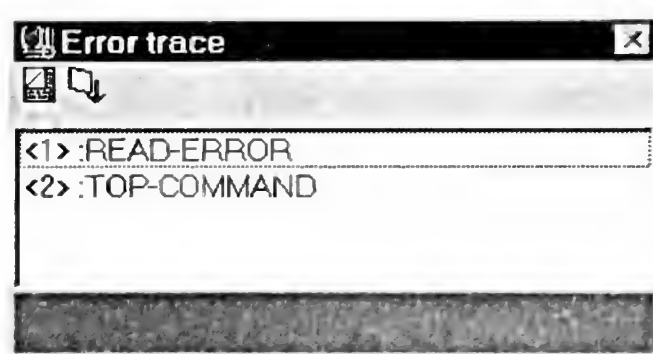


Рис. 7.16. Диалоговое окно Error trace

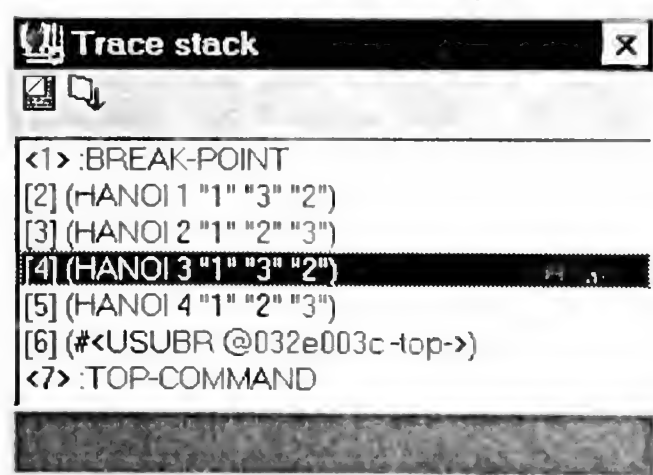


Рис. 7.17. Диалоговое окно Trace Stack

Числа [2], [3], ... – номера элементов в стеке.

Чтобы получить подробную информацию относительно элемента в стеке следа, выделите элемент и щелчком правой кнопки мышки вызовите контекстное меню.

Активные элементы, доступные в контекстном меню, зависят от типа выбранного элемента стека.

Контекстное меню включает следующие команды:

- **Inspect** (Проверить) – вызов диалогового окна проверки **Inspect** (Проверить) выделенного элемента стека;
- **Print** (Печатать) – отображение в окне консоли выделенного элемента стека в виде списка. Например, выделенный на рис. 7.17 элемент будет отображен следующим образом:

```
'(HANOI 3 "1" "3" "2")
```

- **Function symbol** (Имя функции) – вызов диалогового окна **Symbol Service** (Обслуживание символов);
- **Copy** (Копировать) – создание копии выделенного элемента трассы стека в системной переменной ***OBJ***;
- **Local variables** (Локальные переменные) – отображение окна **Frame Binding** (Рамка связывания) для просмотра значений локальных переменных во время выполнения функции;
- **Source position** (Исходная позиция) – отображение текстового окна с исходным текстом с текущей позиции выделенного элемента, если он там имеется, и наоборот;
- **Call point source** (Вызвать исходную точку) – показ позиции выделенного элемента.

Допустим, первая строка программы выглядит следующим образом:

```
(DEFUN PRIVET (/ IMY)
```

Отслеживать значения локальной переменной **IMY** можно во время выполнения функции (**PRIVET...**) в окне **Frame binding** (Рамка связывания), которое представлено на рис. 7.19. Для этого необходимо выбрать пункт **Local variables** (Локальные переменные) контекстного меню выделенного элемента окна трассы стека **Trace Stack**.

Если щелкнуть правой кнопкой мышки в окне **Frame binding**, Visual LISP отобразит контекстное меню.

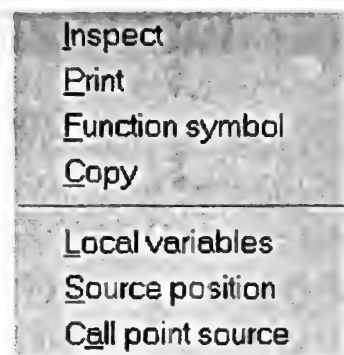


Рис. 7.18. Контекстное меню выделенного элемента окна трассы стека

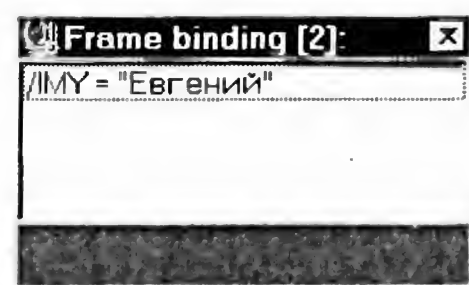


Рис. 7.19. Окно **Frame binding**

Меню содержит следующие пункты:

- **Inspect** (Проверить) – вызов диалогового окна для просмотра выделенного значения;
- **Print** (Печать) – отображение выделенного значения в окне консоли;
- **Symbol...** (Символ) – вызов диалогового окна для обслуживания выделенного символа.
- **Copy** (Копировать) – создание копии выделенного значения в системной переменной ***OBJ***;
- **Add to Watch** (Добавить для наблюдения) – добавление выделенной переменной в окно **Watch** (Окно наблюдений).

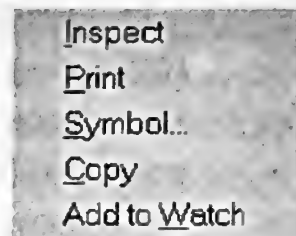


Рис. 7.20.

Контекстное меню
окна *Frame binding*

Когда выполнение программы прерывается из-за ошибки, выберите пункт **Error Trace** (Трассировка ошибки) падающего меню **View** (Просмотр) для просмотра состояний функциональных вызовов до прерывания.

Инструментальная панель в окне стека **Trace** (Трассировка) содержит две кнопки:

- **Refresh** (Освежить) – освежить содержимое окна стека **Trace** (Трассировка);
- **Copy** (Скопировать) – скопировать содержимое окна в окно **Trace** (Трассировка) или файл регистрации.

В процессе отладки для упрощения доступа к различным возможностям отладчика очень часто используют диалоговое окно **Symbol Service** (Обслуживание символов). Чтобы открыть диалоговое окно **Symbol Service**, необходимо:

- выделить группу символов в текстовом окне редактора или в окне консоли;
- выбрать пункт **Symbol Service** (Обслуживание символов) падающего меню **View** (Просмотр) главного меню Visual LISP или щелкнуть по кнопке **Symbol Service** (Обслуживание символов) на инструментальной панели **Debug** (Отладка).

На экране появится диалоговое окно ввода имени символа **Enter symbol name:** (Ввод имени символа), показанное на рис. 7.21. Выделенная группа символов автоматически появляется в текстовом поле раскрывающегося списка в окне ввода имени символов **Enter symbol name:** (Введите имя символа);

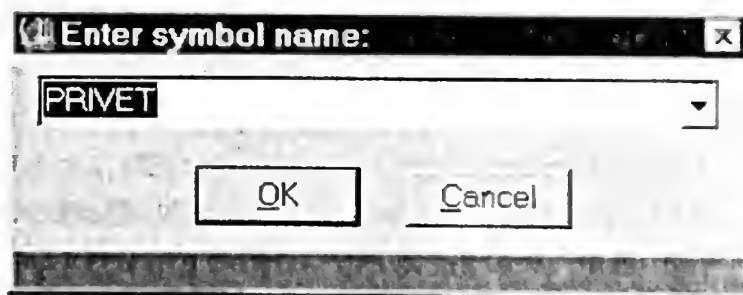


Рис. 7.21. Окно ввода имени символа

- щелкнуть по кнопке **OK** (Okay – хорошо).

На экране появится диалоговое окно **Symbol Service** (Обслуживание символов), показанное на рис. 7.22.

Диалоговое окно **Symbol Service** (Обслуживание символов) содержит:

- инструментальную панель;
- текстовое поле **Name** (Имя) для ввода или изменения символов;
- текстовое поле **Value** (Значение), в котором отображается значение символа или его начальная подстрока;
- раздел **Flags** (Флажки) для включения/выключения атрибутов символа.

Значение отображаемого символа можно изменить путем ввода нового значения в текстовом поле **Value** (Значение) и выбора кнопки **OK**. Visual LISP вычислит и присвоит новое значение символу.

Когда флажок **Protect assign** (Защитить присвоение) включен, поле **Value** (Значение) используется только для отображения значения.

Инструментальная панель **Symbol Service** (Обслуживание символов) содержит следующие кнопки:

- **Watch** (Понаблюдать) – добавить символы в окно **Watch** (Наблюдений);
- **Inspect** (Проверить) – просмотреть и проверить значения символа;
- **Show Definition** (Показать определение) – открыть окно текстового редактора, содержащего функцию, и выделить ее;
- **Help** (Помощь) – отображение информации из файла **Help Visual LISP**, если символ является именем встроенной функции.

Диалоговое окно **Symbol Service** (Обслуживание символов) позволяет задавать атрибуты символам с помощью следующих переключателей:

- **Trace (TR)** (Отслеживать) – отслеживание указанной пользователем функции, имя которой введено в текстовое поле **Name** (Имя). Используется только с именем функции;
- **Protect assign (PA)** (Защитить присвоение) – защита от изменения значения символа. Например, символ **PI** – **Protect assign** – защищенный символ. Все символы, которые являются именами встроенных функций AutoLISP, защищены от изменения по умолчанию;

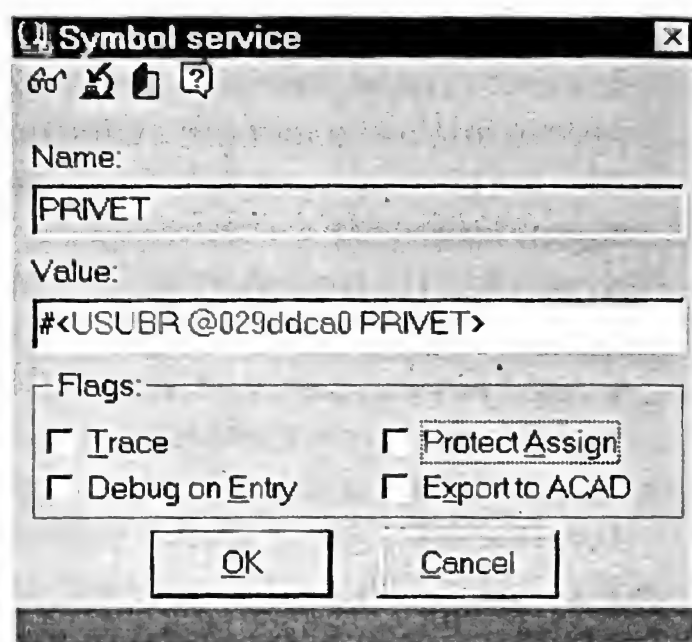


Рис. 7.22. Диалоговое окно **Symbol Service** (Обслуживание символов)

- **Debug on Entry** (Отладка на входе) – установка точки прерывания при каждом вызове. Действует только во время выполнения функции, но не во время ее загрузки или определения. Обратите внимание, что Visual LISP игнорирует переключатель **Debug on Entry** для всех SUBR, EXSUBR и EXRXSUBR символов;
- **Export to ACAD (EA)** (Экспорт в AutoCAD) – определение функции, связанной с символом, как внешней подпрограммы AutoLISP;

Для облегчения отладки используется специализированное средство – **Inspector** (Проверка), которое позволяет просматривать, исследовать и изменять объекты AutoCAD и AutoLISP. С помощью этого эффективного и доступного в использовании средства можно осуществить просмотр:

- любых объектов, доступных в AutoLISP (например, списков, чисел, строк и переменных);
- нарисованных объектов AutoCAD;
- выделенных наборов примитивов AutoCAD;
- сложных структур данных.

Inspector (Проверка) создает окно для любого объекта, который подвергается проверке.

Окна **Inspector** имеют одинаковый вид и содержат заголовок окна, например, **Inspect: STR** (рис. 7.23), строку объекта (под заголовком) и список элементов объекта (возможно, пустой).

Допустим, вы запустили на выполнение файл PRIVET1W.LSP.

На запрос в командной строке:

Command:

Введите Ваше имя ! - - -

ввели, например, имя *Евгений*. Затем вы вернулись в окно текстового редактора и выделили переменную IMY. После этого решили вызвать окно **Inspect:** (Проверить). Для этого щелкнули по кнопке **Inspect** (Проверить) – кнопке с изображением на ней микроскопа – или выбрали пункт **Inspect...** (Проверить) падающего меню **View** (Просмотр). В зависимости от типа выделенной переменной на экране появится соответствующее окно проверки (в нашем случае окно **Inspect: STR** (Проверка строковой константы), представленное на рис. 7.23).

Заголовок окна **Inspect:** показывает тип проверяемого объекта, в данном случае **STR** (String – строковая константа).

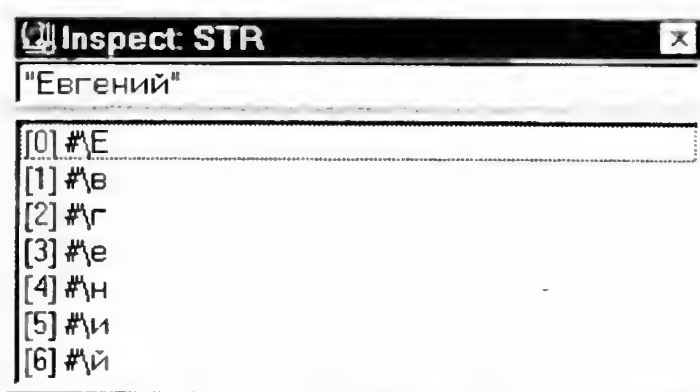


Рис. 7.23. Окно проверки строковой константы
Inspect: STR

Строка объекта (**Object line**) показывает напечатанное представление заданного объекта.

Список элементов (**Element list**) отображает компоненты заданного объекта. Размер и содержание списка элементов меняются в зависимости от типа объекта. Каждый элемент списка имеет имя (оно заключено в квадратные скобки) и содержание.

Строка объектов и строки списка элементов имеют их собственные связанные контекстные меню.

Для различных типов объектов AutoLISP используются следующие виды списков элементов:

- **INT Inspector** (Проверка целого числа), который представляет целое число в различных системах счисления;
- **REAL Inspector** (Просмотр действительного числа) – этот список элементов пустой;
- **STRING Inspector** (Просмотр строковой константы), который содержит символы строковой константы в определенной последовательности. Каждый элемент списка может быть осмотрен как целое число;
- **Symbol Inspector** (Окно просмотра символов) – просмотр списка элементов символа содержит 3 элемента: значение, имя и атрибуты;
- **LIST Inspector** – список элементов списка AutoLISP содержит 2 элемента: начальный элемент списка (эквивалент CAR), список без первого элемента (эквивалент CDR) для всех случаев, когда CDR не NIL.
- **File Inspector** (Просмотр файла) – список элементов файла, который содержит имя файла и параметры его открытия;
- **SUBR, EXSUBR и USUBR** инспектора, которые содержат имя функции, определенное функцией (**DEFUN...**) или указанное во времени загрузки;
- **ENAME Inspector** (Проверка рисунков объекта) – просмотр списка подписков, в которых сгруппированы по функциональному назначению все данные о примитиве AutoCAD – как геометрические, так и общие: слой, цвет и т. д. Подписки отличаются друг от друга по специальным кодам формата **DXF** (**D**rawing **e**Xchange **F**ormat – формат обмена рисунками). Каждый подписок имеет две части: код DXF и данные.
- **PICKSET Inspector** (Просмотр выбранных объектов) – просмотр списка выбранных объектов AutoCAD.

Чтобы просмотреть объект, выделите имя объекта в любом контексте, выберите пункт **Inspect** (Проверить) падающего меню **View** (Просмотр) главного меню Visual LISP или щелкните кнопку **Inspect** на инструментальной панели **Debug** (Отладка). Пункт **Inspect** (Проверить) также

доступен из ряда контекстных меню и диалоговых окон, например, окна **Symbol Service** (Обслуживание символов).

Для того чтобы просмотреть определение функции (**PRIVET...**), выделите имя в окне текстового редактора, затем щелкните кнопку **Inspect** (Проверить) на инструментальной панели **Debug** (Отладка). На экране появится окно проверки функции пользователя **Inspect: USUBR**, представленное на рис. 7.24.

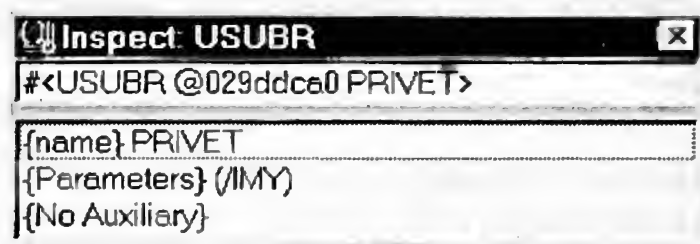


Рис. 7.24. Окно проверки функции *Inspect: USUBR*

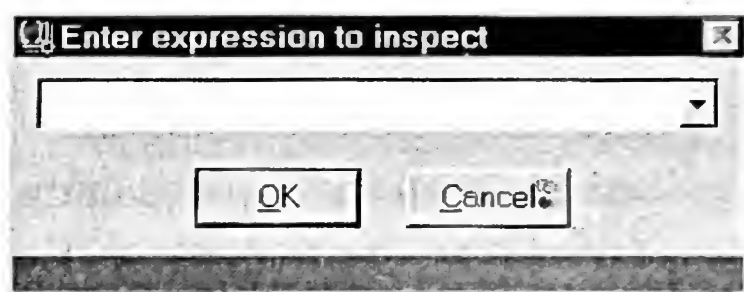


Рис. 7.25. Окно ввода выражения для проверки

Если вы вызываете пункт **Inspect**, не выбирая имени объекта, Visual LISP позволяет определить объект в окне ввода выражения для проверки **Enter expression to inspect** (Введите выражение для проверки) (рис. 7.25).

Введите в текстовом поле раскрывающегося списка имя объекта или выражение, которое хотите проверить, затем щелкните по кнопке **OK** (Окay – хорошо), чтобы открыть окно **Inspector** (Проверка), или нажмите кнопку **Cancel** (Отменить) для отмены действия. В раскрывающемся списке Visual LISP сохраняет 15 последних введенных выражений. Если интересующий вас объект или выражение находится в раскрывающемся списке, то для его проверки достаточно дважды щелкнуть по нему мышкой.

В окне текстового редактора нельзя просмотреть выбранные выражения, которые длиннее 256 символов. При выборе строки, количество символов в которой превышает 256, компьютер запросит ввести имя объекта.

Если вы зададите объект или выражение, которое Visual LISP не может обработать, на экране появится стандартное сообщение об ошибках AutoLISP. В таком случае вы можете исправить выражение в диалоговом окне и попробовать обработать его еще раз.

Ошибки, ставшие результатом обработки объекта, который вы ввели, не могут быть исследованы из вложенного уровня прерывания, потому что все прерывания во время обработки заблокированы. При желании исследовать ошибку выберите пункт **Error Trace** (Трассировка ошибки) падающего меню **View** (Просмотр) главного меню Visual LISP или скопируйте выражение с подсказки консоли и нажмите клавишу **Enter** (Ввод).

Закрыть все окна **Inspector** (Проверка) можно, если выбрать пункт **Inspectors** (Проверки) всплывающего меню **Close Windows** (Закрыть окна) падающего меню **Window** (Окно) главного меню Visual LISP.

Диалоговые окна **Inspector** (Проверка) имеют контекстные меню, зависящие от просматриваемых данных.

Контекстное меню строки объекта появляется после нажатия комбинации клавиш **Alt+O** или щелчка правой кнопкой мышки по строке объекта диалогового окна и может содержать следующие команды:

- **Symbol Service** (Обслужить символы) – вызывает диалоговое окно обслуживания символов;
- **Print** (Напечатать) (**Alt+P**) – отображает объект в окне консоли;
- **Pretty print** (Структурная печать) – форматирует и отображает объект в окне консоли;
- **Copy** (Скопировать) – копирует объект в системную переменную ***OBJ***;
- **Log** (Зарегистрировать) – копирует текущее содержание окна **Inspector** в окно **Trace frame** (Трассировка рамки);
- **Update** (**Alt+U**) (Модифицировать) – модифицирует окно **Inspector** (Проверка), чтобы показать последнее состояние просматриваемого объекта.

После того, как щелчком правой кнопки мышки вы подсветите элемент списка, появится контекстное меню элементов списка, которое содержит следующие пункты:

- **Inspect** (Проверить) (**Alt+I**) – вызов окна **Inspector** для данного элемента списка;
- **Descend** (**Alt+D**) – вызов окна **Inspector** для данного элемента списка и закрывание текущего окна **Inspector**;
- **Copy** (Копировать) – копирование значения просматриваемого элемента в системную переменную ***OBJ***;
- **View source** (Просмотр источника) – активизация окна текстового редактора, которое содержит выбранный текст. Если окно было загружено с консоли, активизируется новое окно текстового редактора.

Иногда необходимо обратиться к некоторой части объекта из программы или из окна консоли Visual LISP. Вы можете, например, копировать значение элемента одного объекта в другой и так далее, если воспользуетесь зарезервированной системной переменной ***OBJ***, которая при просмотре структуры данных может использоваться как временная область памяти. Из окна проверки **Inspector** (Проверка) можно задавать или изменять значение этой системной переменной.

Чтобы поместить значение объекта в системную переменную ***OBJ***, выделите этот объект, щелкните правой кнопкой мышки в окне проверки **Inspector** и выберите пункт **Copy** (Копировать).

Рассмотрим окна проверки **Inspect**: для различных типов данных Visual LISP, которые используются наиболее часто.

Окно **Inspect: INT** (Проверить целое число), представленное на рис. 7.26, показывает список возможных представлений целого числа: двоичного, восьмеричного, десятичного, шестнадцатеричного и символьного (ASCII код).

Окно **Inspect: REAL**, представленное на рис. 7.27, не содержит списка элементов (**Element list**).

Окно **Inspect:**, представленное на рис. 7.28, позволяет просматривать имя объекта и содержит следующие элементы:

[value] – значение символа;

{name} – имя символа, которое всегда является строкой;

<flags...> {} – атрибуты символа: **PA** (**P**rotect **A**ssign) (Защитить присваивание); **TR** (**T**race) (Трассировка); **DE** (**D**ebug) (Отладка); **EA** (**E**xport AutoCAD) (Экспортировать в AutoCAD).

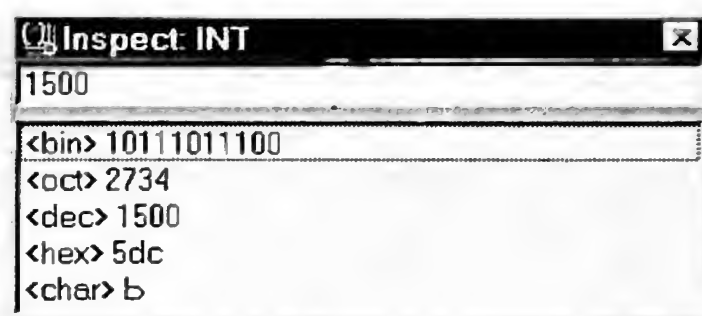


Рис. 7.26. Окно представления целого числа



Рис. 7.27. Окно представления действительного числа



Рис. 7.28. Окно проверки имени объекта

7.2. Создание приложений в среде Visual LISP

Visual LISP имеет эффективное средство для создания автономного приложения **Application Wizard** (Мастер приложений). Следует иметь в виду, что приложение может быть создано только после того, как оно полностью отлажено.

Чтобы начать работу **Application Wizard** (Мастер приложений), выберите пункт **New Application Wizard** всплывающего меню пункта **Make Application** (Создать приложение) падающего меню **File** (Файл) главного меню Visual LISP. На экране появится диалоговое окно **Create new Make-Application File** (Создать новый файл приложения), представленное на рис. 7.29.

В диалоговом окне выберите путь и введите имя файла. В нашем примере имя файла совпадает с именем функции. К имени файла мастер автоматически добавляет расширение MKP. Нажмите клавишу **Enter** (Ввод) или щелкните мышкой по кнопке **Сохранить** (Save).

Рассмотрим порядок создания приложения с помощью **Wizard** (Мастера).

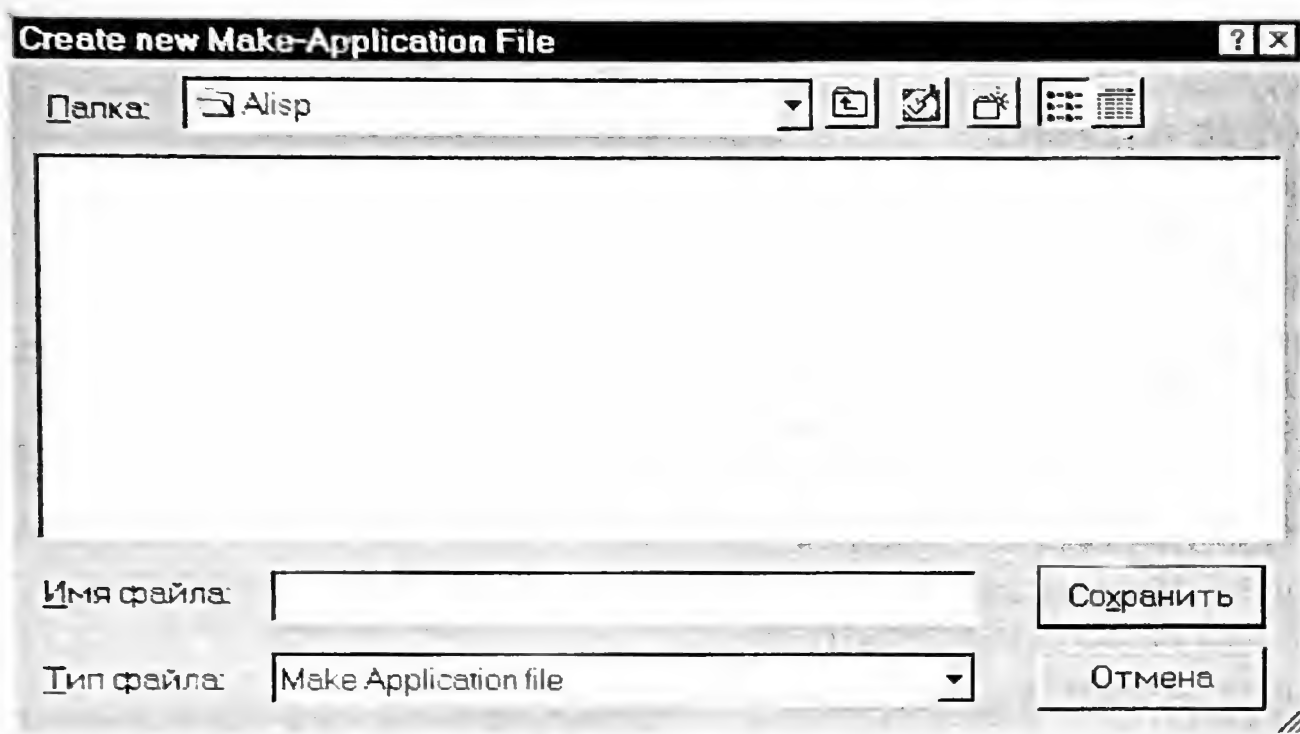


Рис. 7.29. Диалоговое окно создания нового файла

На первом шаге в разделе **Select the application type** (Выбрать тип приложения) выбирается тип приложения (см. рис. 7.30):

ARX Standard – Visual LISP создает один модуль ARX, который может быть загружен и выполнен в AutoCAD. Этот модуль наряду с программой

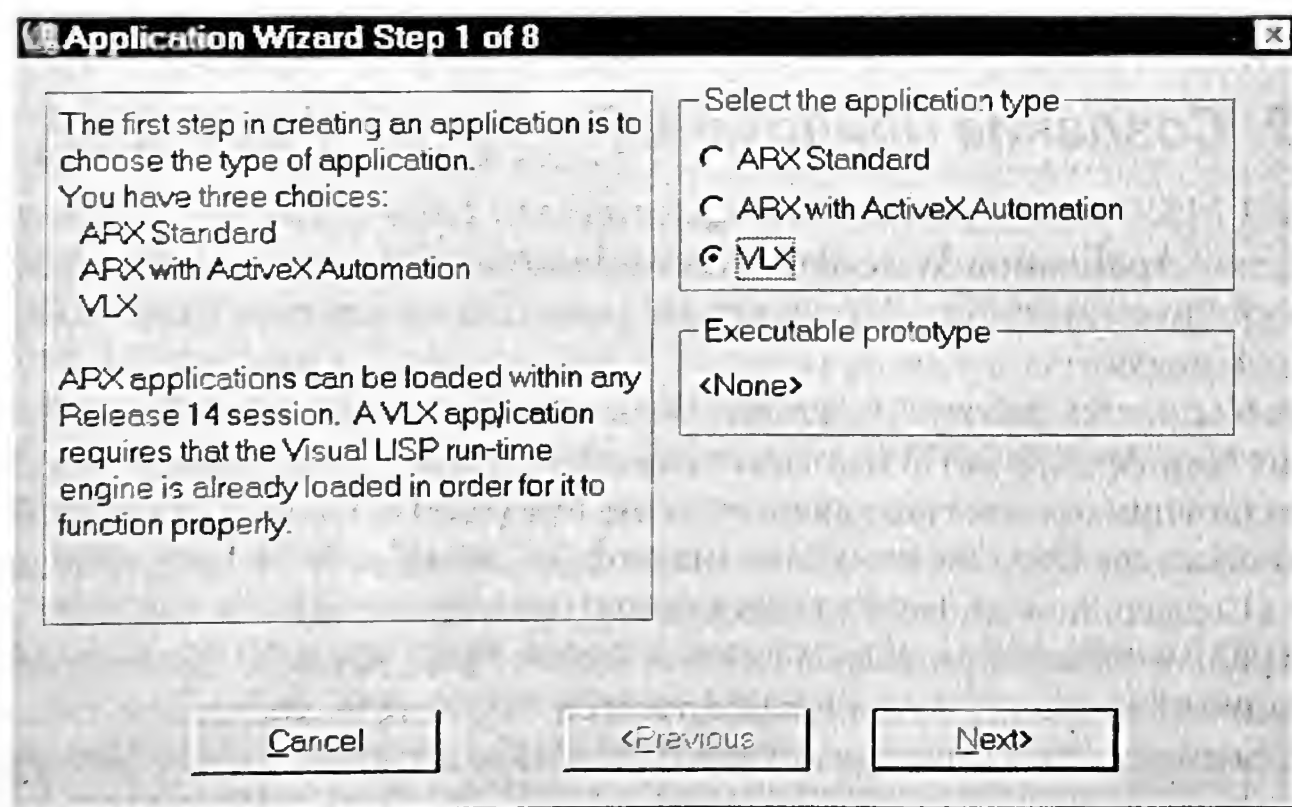


Рис. 7.30. Диалоговое окно Application Wizard на первом шаге

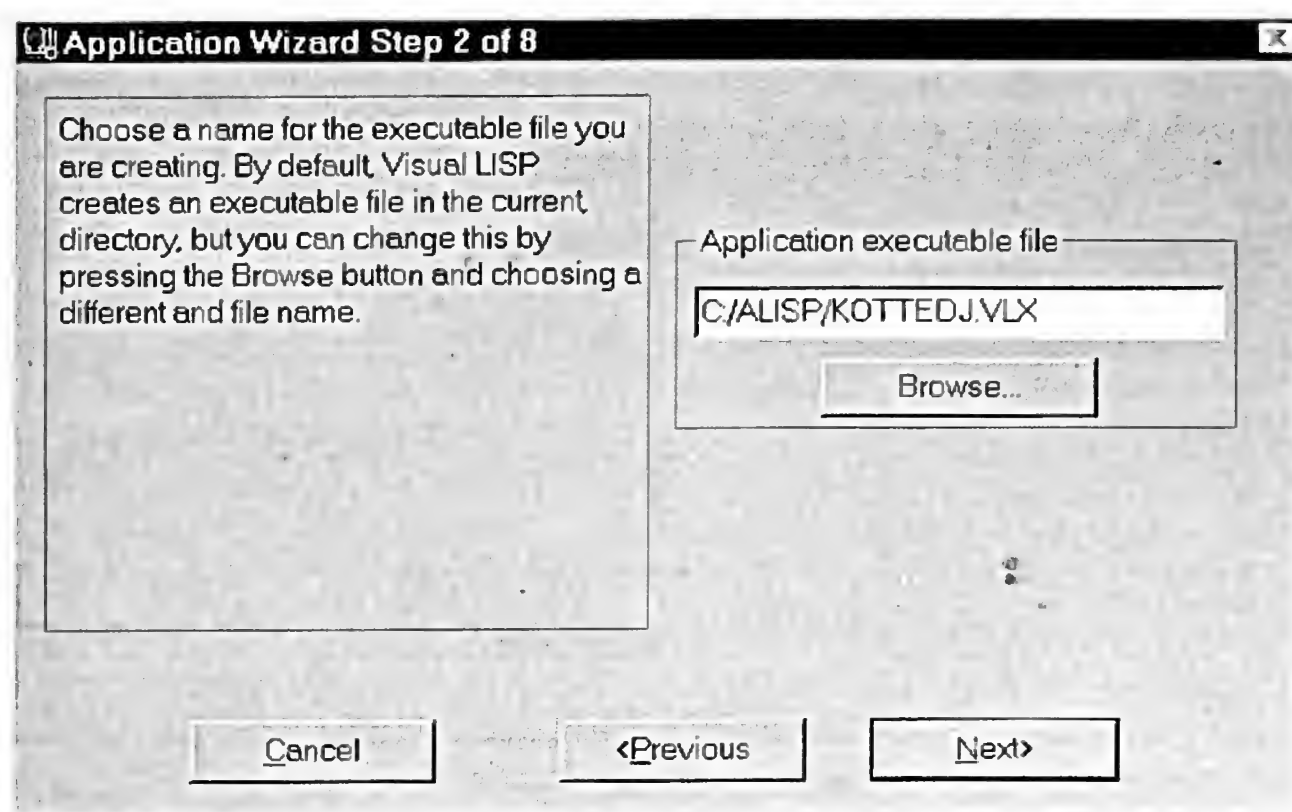


Рис. 7.31. Диалоговое окно Application Wizard на втором шаге

и DCL (Dialog Control Language – язык управления диалогом) файлами включает копию **Run-Time System (RTS)** Visual LISP;

ARX with ActiveX Automation – Visual LISP также создаст один модуль ARX, подобно **Standard**, за исключением того, что в модуле используется методология ActiveX. Если программа использует в приложении методологию ActiveX, то для создания приложения вы должны выбрать именно этот режим. В случае выбора типа **Standard** программа не будет выполняться в AutoCAD;

VLX – содержит приложение и все файлы обеспечения, однако не включает RTS Visual LISP. Поскольку RTS не включен, размер модуля VLX меньше размера модуля ARX и для всех модулей VLX можно использовать один RTS. Для продолжения процесса создания приложения щелкните по кнопке **Next** (Следующий).

На втором шаге определяется имя исполняемого файла приложения **Application executable file** (Исполняемый файл приложения) (рис. 7.31).

По умолчанию Visual LISP создаст исполняемый файл в текущем каталоге. Можно также задать полное имя пути или щелкнуть по кнопке **Browse** (Пролистать) и выбрать нужное имя файла.

Если необходимо вернуться к предыдущему шагу Application Wizard и что-либо изменить, щелкните по кнопке **Previous** (Предыдущий). Для перехода к следующему шагу щелкните по кнопке **Next** (Следующий).

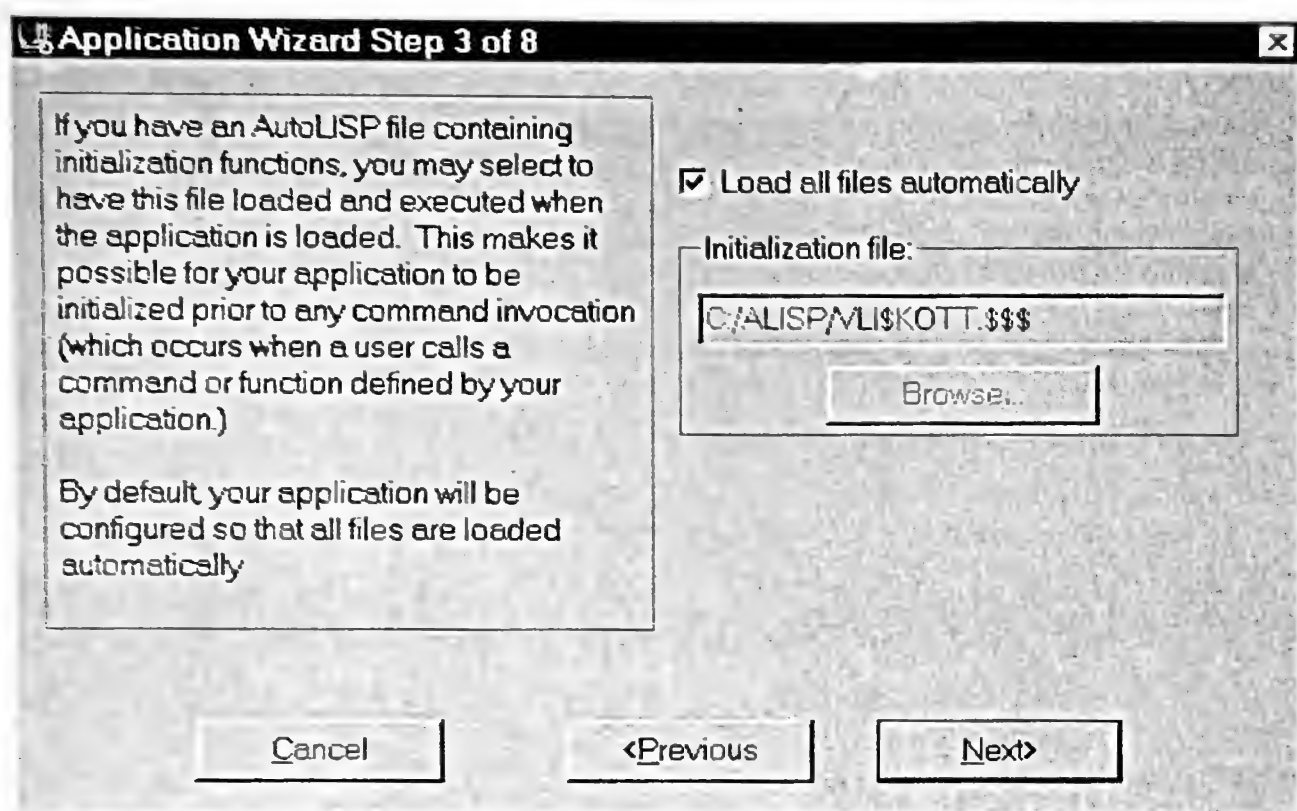


Рис. 7.32. Диалоговое окно *Application Wizard* на третьем шаге

На третьем шаге производится инициализация файлов **Initialization file** (Файл инициализации) (рис. 7.32).

По умолчанию приложение будет сконфигурировано таким образом, что все файлы загрузятся автоматически. Visual LISP создаст файл запуска VLISPPREFIX.SSS, где PREFIX – первые четыре символа исполняемого файла, имя которого было задано на втором шаге. Файл запуска загружает все файлы FAS приложения. Не забудьте, что в файлы FAS можно включить функции инициализации.

При выборе файла инициализации (типа VLARTS.LSP) выключите флажок **Load of files automatically** (Загрузить файлы автоматически) и щелкните по кнопке **Browse** (Пролистать), чтобы ввести имя файла инициализации, который создан. Данный файл будет загружен при запуске.

На четвертом шаге производится выбор файлов, которые должны быть включены в приложение (рис. 7.33).

У пользователя есть возможность выбрать исходные файлы AutoLISP (с расширением LSP), скомпилированные LISP файлы (с расширением FAS) либо файлы проекта Visual LISP (с расширением PRJ) или любую их комбинацию.

При выборе файлов проекта в модуль вывода включаются файлы FAS всего проекта.

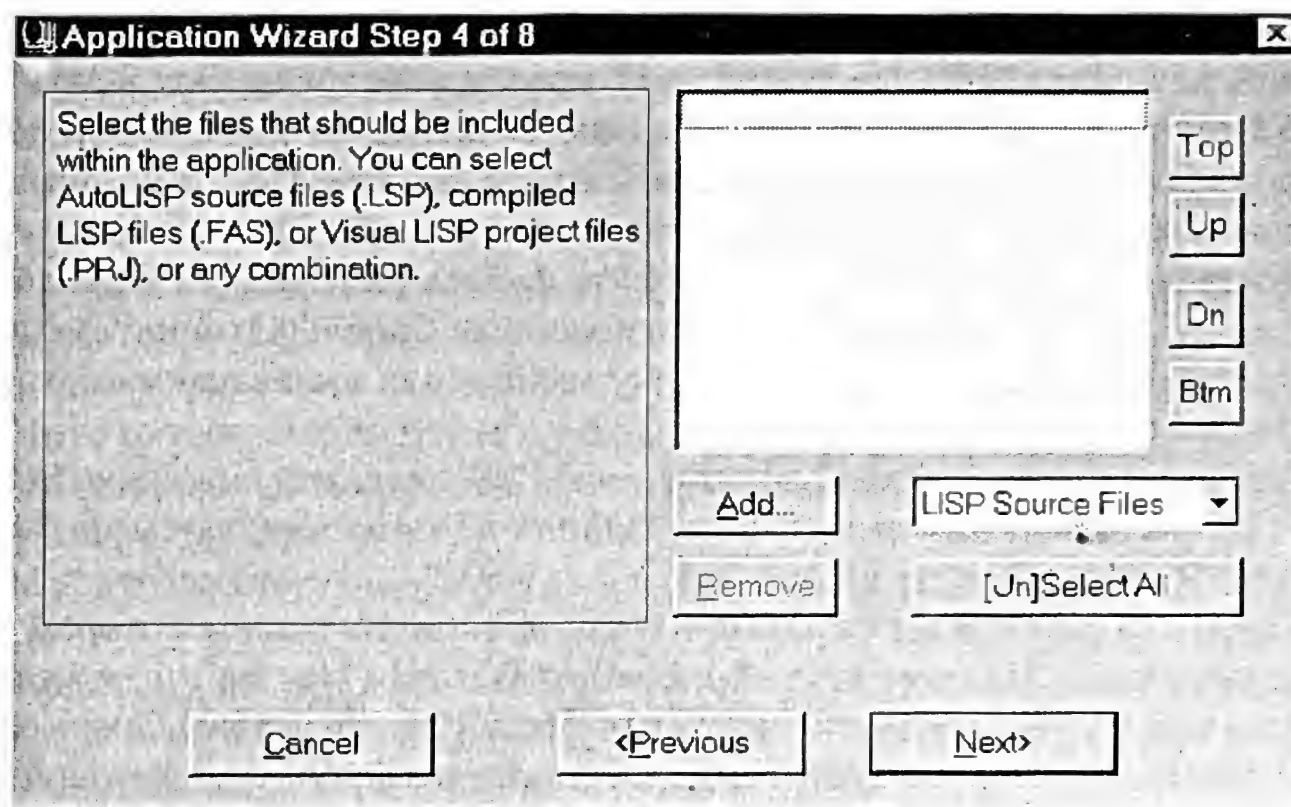


Рис. 7.33. Диалоговое окно Application Wizard на четвертом шаге

В случае выбора исходных файлов AutoLISP Visual LISP компилирует файлы, которые входят в приложение.

Для того чтобы выбрать тип файла, который необходимо включить в приложение, раскройте список и щелкните по кнопке Add (Добавить).

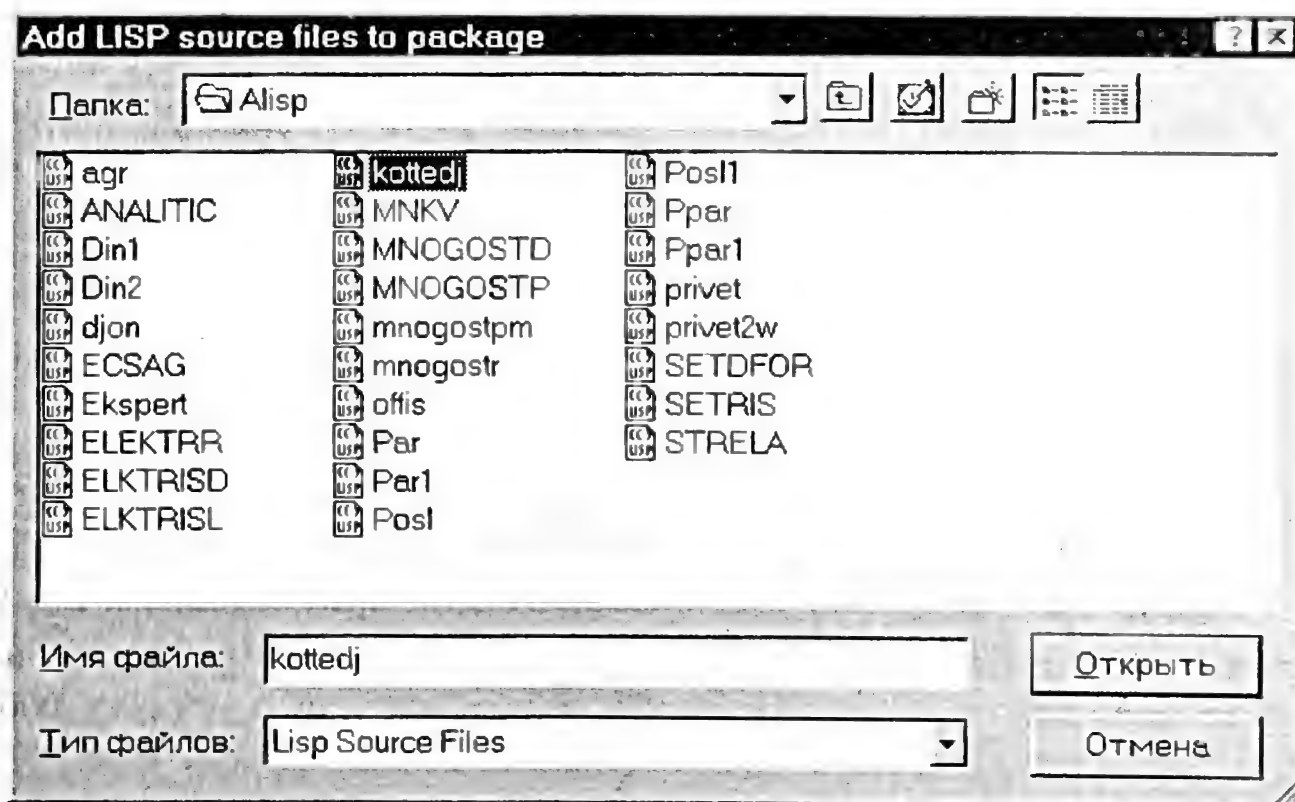


Рис. 7.34. Диалоговое окно добавления исходных LISP файлов в пакет

На экране появится диалоговое окно **Add LISP source files to package** (Добавить исходные LISP файлы в пакет), представленное на рис. 7.34.

В этом диалоговом окне можно выбрать только один файл. Чтобы добавить выбранный файл в приложение, можно дважды щелкнуть по нему мышкой или нажать кнопку **Открыть** (Open) (рис. 7.35).

Когда вы хотите удалить или заменить файлы, включенные в приложение, выделите эти файлы и щелкните по кнопке **Cancel** (Отменить). Можно также использовать правую кнопку мышки для вызова соответствующего контекстного меню.

Для замены всех файлов LSP на файлы FAS щелкните по кнопке **Select All** (Выбрать все) для выбора всех файлов приложения, затем нажмите кнопку **Remove** (Удалить).

Visual LISP загружает файлы приложения в том порядке, в котором они перечисляются. На этом шаге **Application Wizard** (Мастер приложения) можно переупорядочить список файлов. Например, функция должна быть определена прежде, чем будет выполнена загрузка функции, в которой она используется. Диалоговое окно **Application Wizard Step 4 of 8** содержит кнопки, которые можно использовать, чтобы переместить файлы в списке. Выберите имя файла, затем щелкните по кнопке:

- **Top** (В начало) – для перемещения в начало списка файлов;
- **Up** (Вверх) – для перемещения в списке файлов на одну позицию вверх;

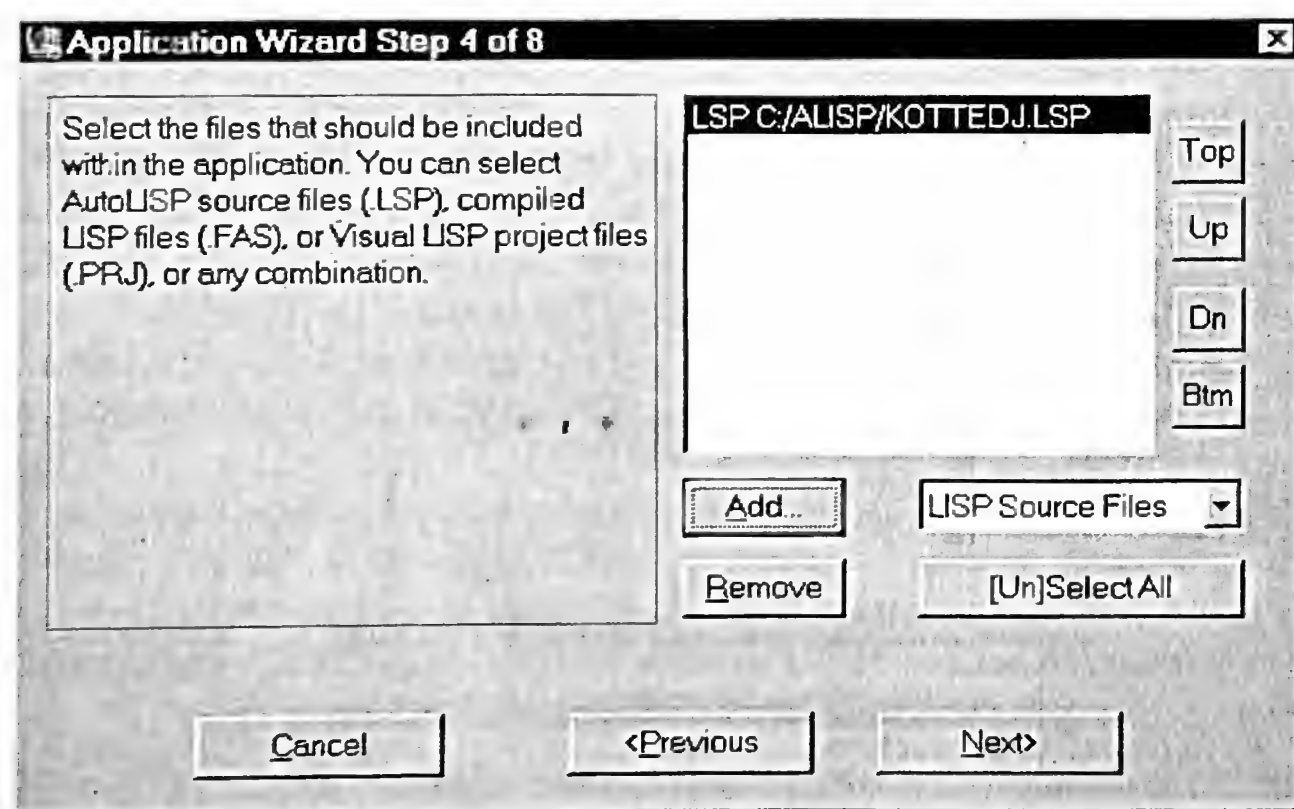


Рис. 7.35. Диалоговое окно с выбранным файлом

- **Dn** (Вниз) – для перемещения в списке файлов на одну позицию вниз;
- **Btm** (В конец) – для перемещения в нижнюю часть списка файлов.

Эти же действия можно выполнить с помощью контекстного меню, которое вызывается щелчком правой кнопки мышки (рис. 7.36).

Контекстное меню содержит следующие пункты:

- **Remove** – удалить из списка;
- **Move up** – переместить на одну позицию вверх в списке файлов;
- **Move to top** – переместить в начало списка файлов;
- **Move down** – переместить на одну позицию вниз в списке файлов;
- **Move to bottom** – переместить в конец списка файлов.

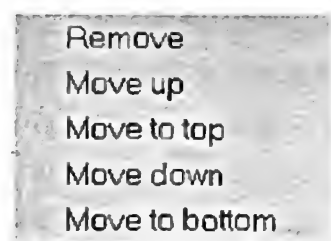


Рис. 7.36.

Контекстное меню перемещения имен файлов в списке

Начиная с 12-й версии, AutoCAD располагает средствами для создания диалоговых окон DCL (Dialog Control Language – язык управления диалогом).

На *пятом шаге* производится включение в приложение файлов DCL (Dialog Control Language – язык управления диалогом) с предварительной компиляцией, однако их можно и не включать.

На *шестом шаге* производится создание External Definitions File (Внешне определенных файлов) (XDF) для функций, определенных в других ARX или ADS приложениях (см. рис. 7.38).

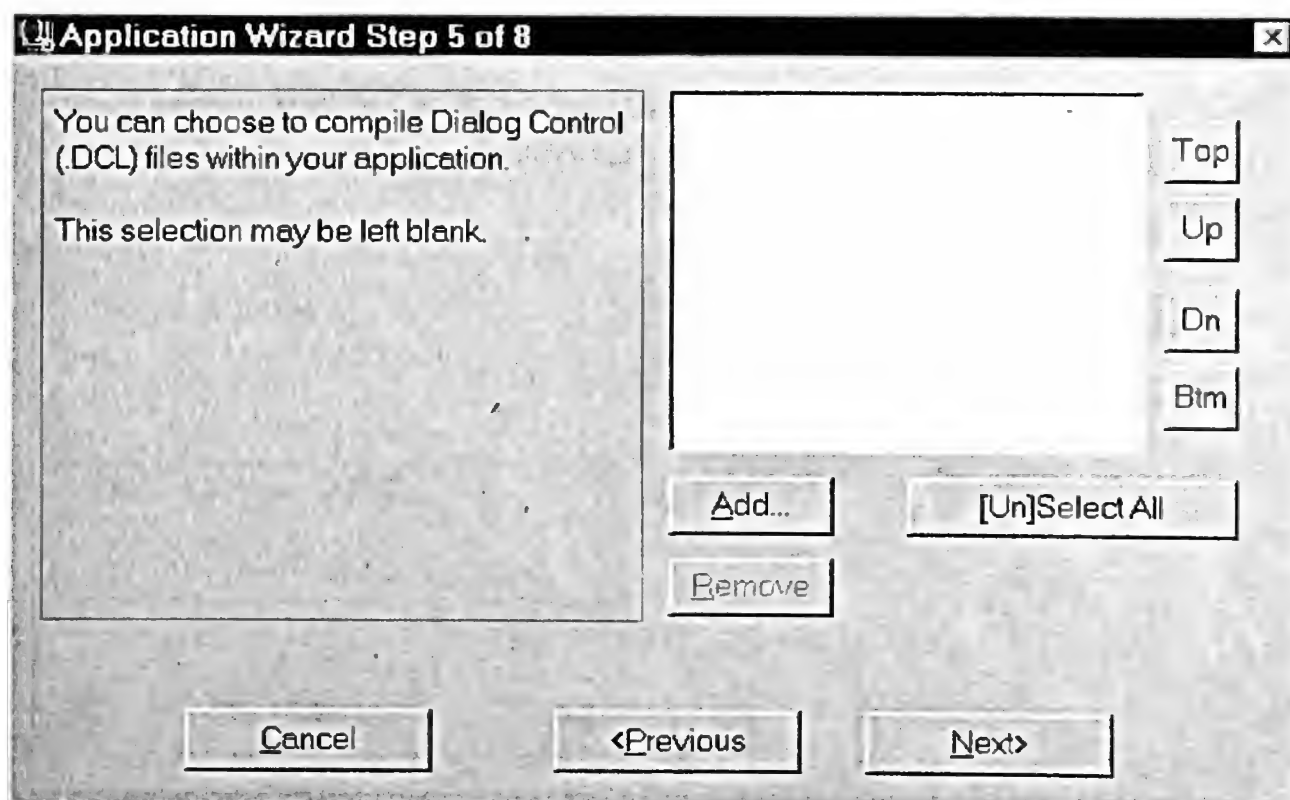


Рис. 7.37. Диалоговое окно Application Wizard на пятом шаге

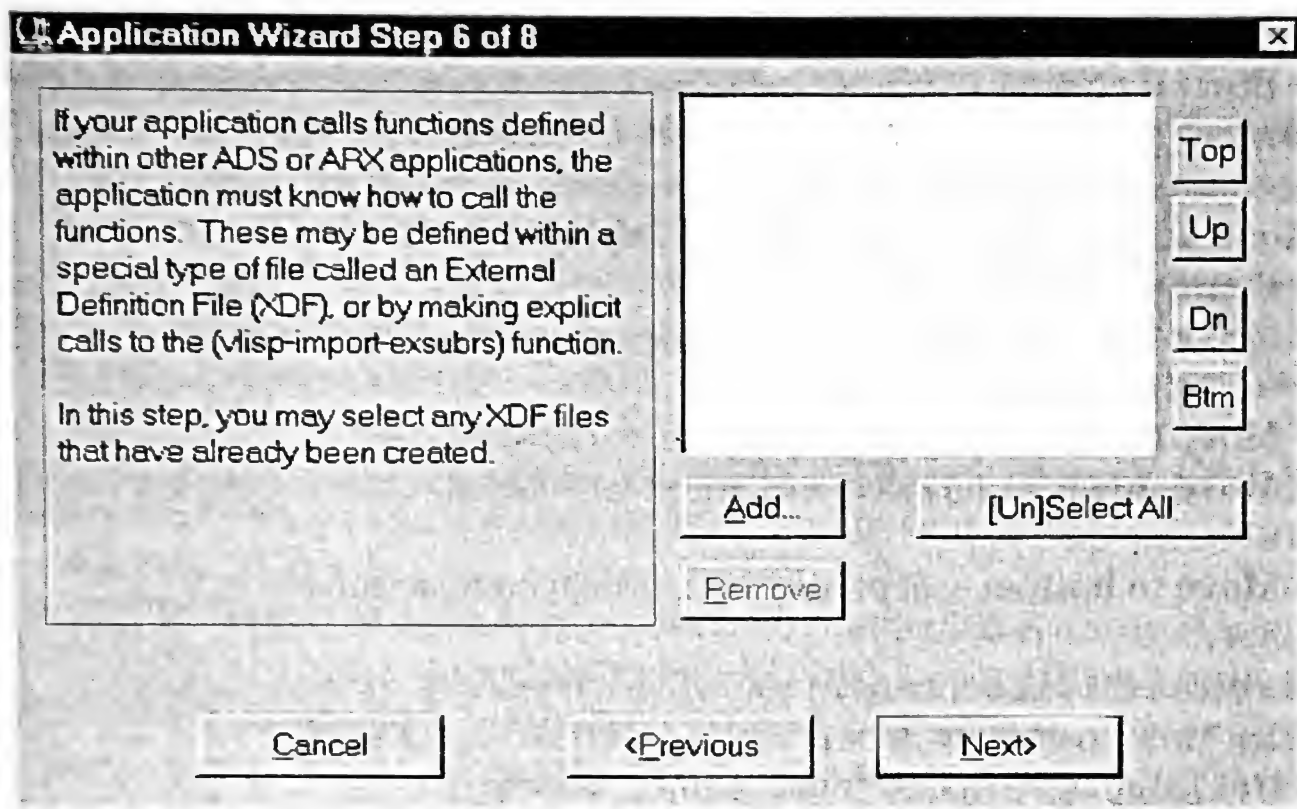


Рис. 7.38. Диалоговое окно Application Wizard на шестом шаге

При этом можно использовать XDF файлы, которые были созданы ранее. На седьмом шаге производится инициализация приложения (во время загрузки или во время первого вызова функции инициализации) (см. рис. 7.39).

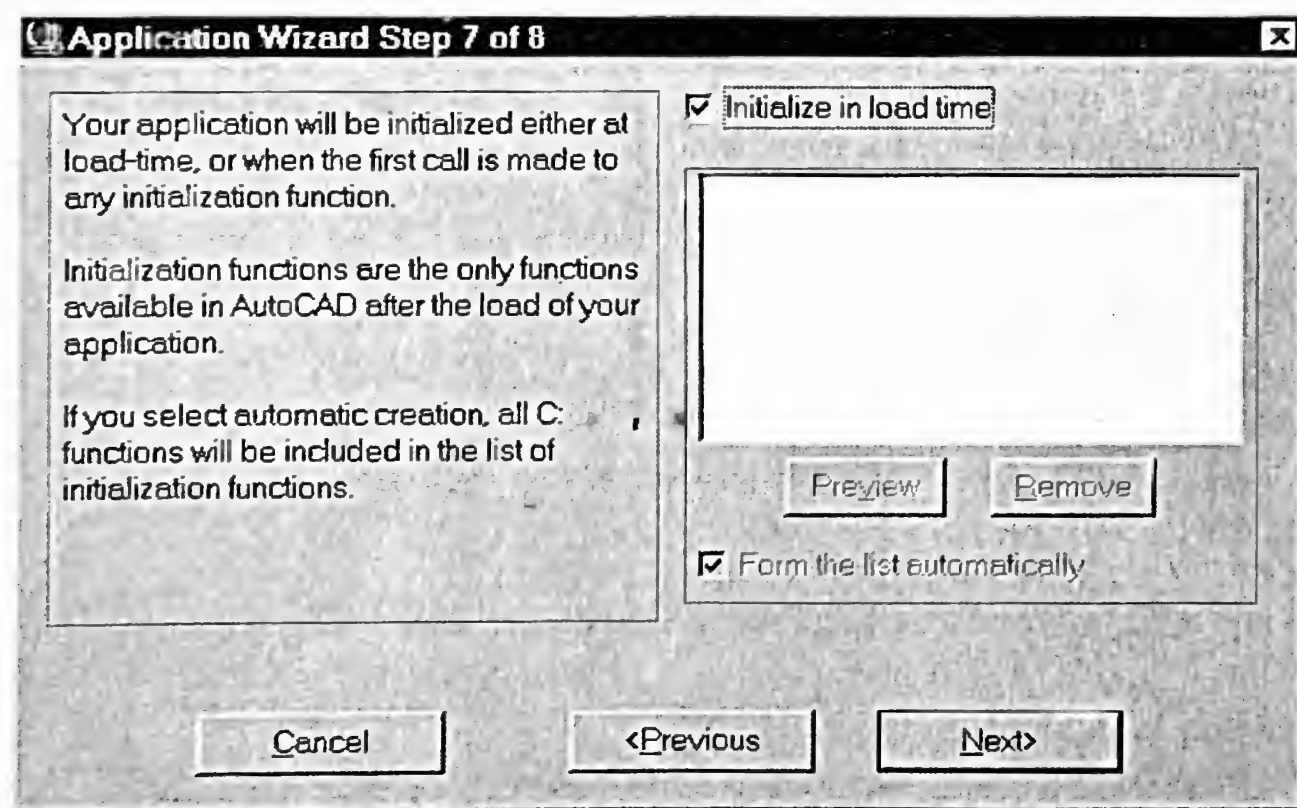


Рис. 7.39. Диалоговое окно Application Wizard на седьмом шаге

Чтобы обеспечить инициализацию создаваемого приложения во время загрузки, включите флажок в окошке **Initialize in load time** (Инициализировать во время загрузки).

Во время первого вызова функций инициализации необходимо создать их список. *Функции инициализации* – это функции, которые предназначены для использования в AutoCAD. К ним относятся все функции с именем C:<имя>. Кроме того, обращение к функции **VL-EXPORT-TO-ACAD** связывает инициализацию приложения.

При включенном в окошке **Form the list automatically** (Создать список автоматически) флажке список функций инициализации будет сформирован автоматически. Чтобы выбрать этот режим, необходимо вначале убрать флажок в окошке **Initialize in load time** (Инициализировать во время загрузки).

Если установлен флажок в окошке **Form the list automatically** (Создать список автоматически), можно с помощью кнопки **Preview** (Предварительный просмотр) просмотреть список функций, который Visual LISP создает.

Когда в окошках **Initialize in load time** (Инициализировать во время загрузки) и **Form the list automatically** (Создать список автоматически) не включены флажки, функции инициализации можно выбирать избирательно. При этом на месте кнопки **Preview** (Предварительный просмотр) появляется кнопка **Add** (Добавить).

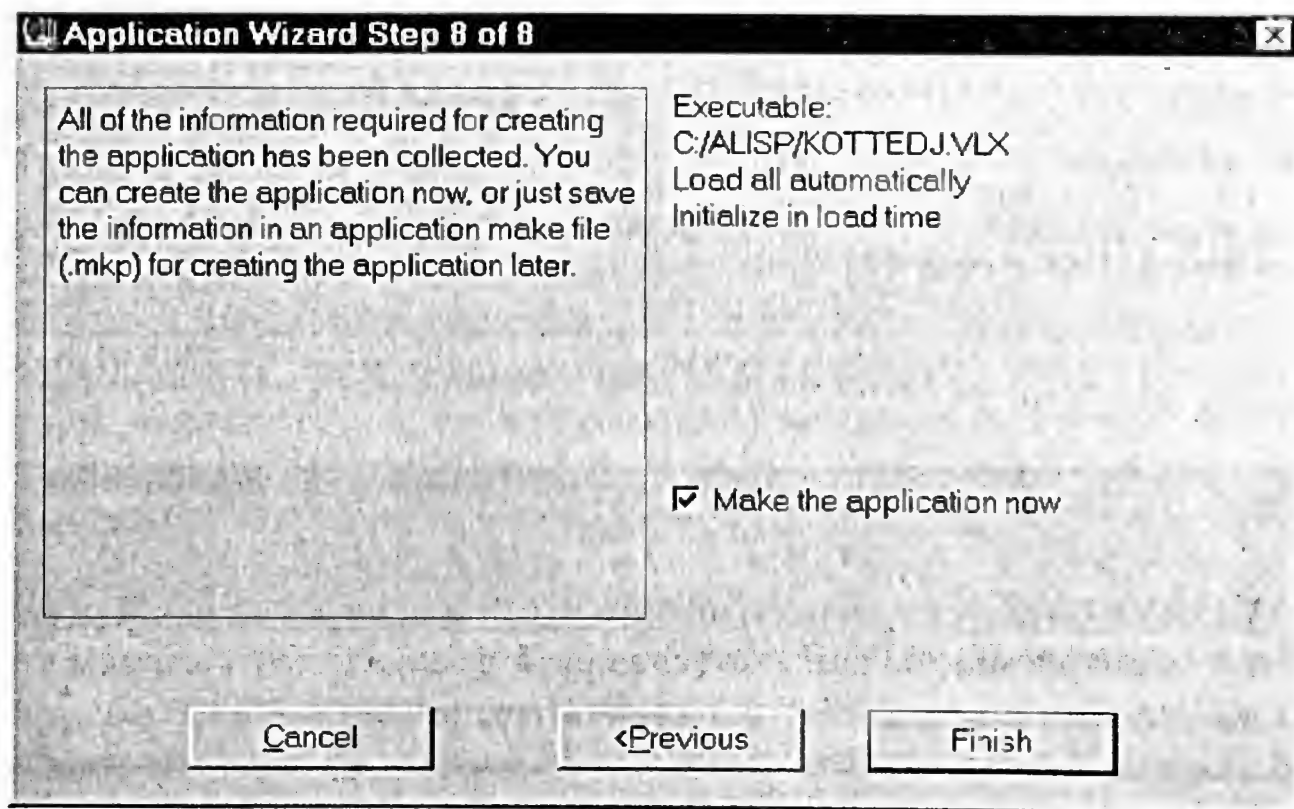


Рис. 7.40. Диалоговое окно Application Wizard на восьмом шаге

На заключительном шаге (рис. 7.41) производится создание приложения или вся необходимая для этого информация сохраняется в MAKE файле. Если щелкнуть по кнопке **Finish** (Закончить), появится окно **Build Output** (Создание выходного файла), в котором отображается информация о завершении процесса компиляции приложения (см. рис. 7.41).

Одновременно в окне консоли появится сообщение о завершении создания приложения, как на рис. 7.41.

При отказе создавать приложение Visual LISP сохраняет информацию в файле с расширением MKP, который можно использовать для этой же цели позднее. Чтобы создать приложение из файла с расширением MKP, необходимо выбрать пункт **Load file** (Загрузить файл) падающего меню **File** (Файл) главного меню Visual LISP и задать имя файла с расширением MKP.

Когда файл с расширением MKP выполняется, Visual LISP автоматически компилирует любые исходные файлы приложения, которые отвечают следующим условиям:

- компилируемая (FAS) версия файла отсутствует;
- компилируемая версия имеется, однако исходный файл после его компиляции изменился;

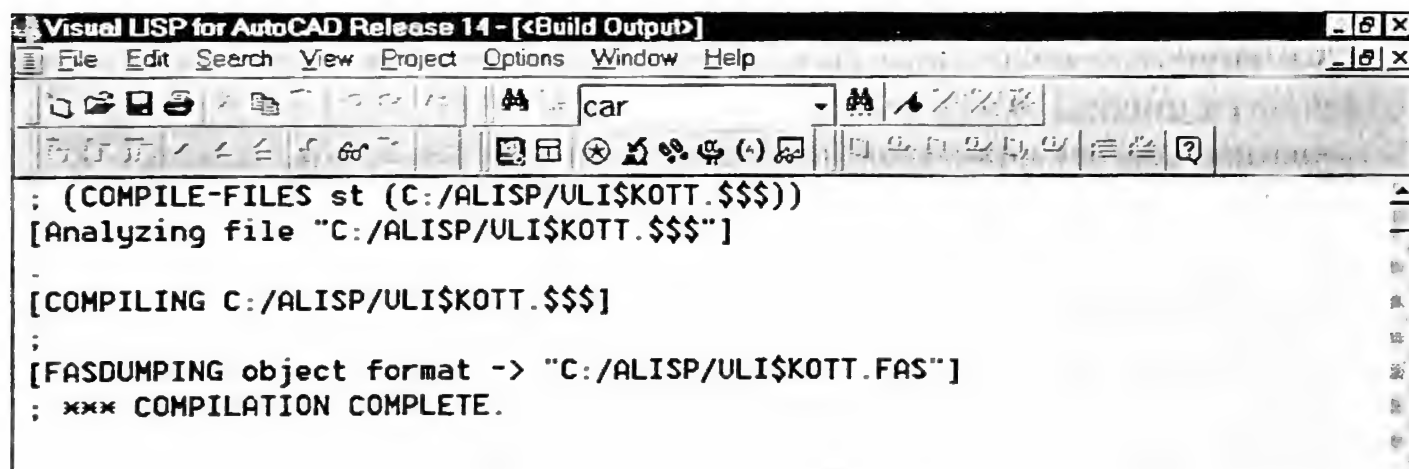


Рис. 7.41. Окно сообщений компилятора Build Output (Создание выходного файла)

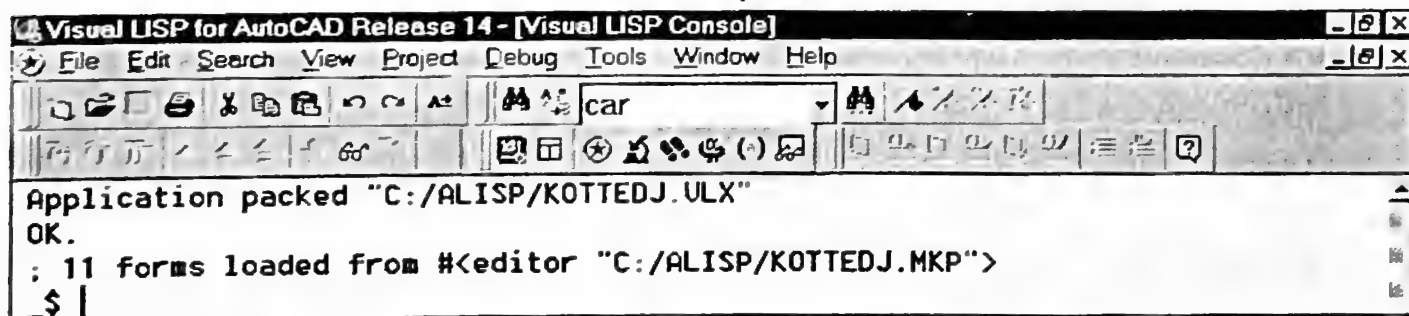


Рис. 7.42. Окно консоли с сообщениями о результатах работы Application Wizard (Мастер приложений)

В результате создается файл, название которого состоит из имени исходного файла с добавлением в начало символов **VLI\$** и расширением **\$\$\$**.

Порядок запуска на выполнение в AutoCAD автономного приложения зависит от того, какой создан модуль – **ARX** или **VLX**.

Чтобы выполнить в AutoCAD приложение **ARX**, загрузите файл **ARX** и вызовите его из подсказки **COMMAND**. Загрузить файл можно с помощью функции (**ARXLOAD...**) или команды **Load Application** (Загрузить приложение) из падающего меню **Tools** (Инструменты) главного меню AutoCAD.

Функция (**ARXLOAD...**) имеет следующий синтаксис:

```
(ARXLOAD <"имя файла">)
```

Для имени файла определите полное имя пути файла **ARX**. Например:

```
(ARXLOAD "C:/ALISP/KOTTEDJ.ARX ")
```

Если загрузка прошла неудачно, то есть имя загружаемой функции не появилось, проверьте, возможно, приложение уже загружено. Команда **ARX** в командной строке AutoCAD позволяет просмотреть список загруженных приложений **ARX**.

Чтобы выгрузить приложение, которое в настоящее время загружено, используйте функцию (**ARXUNLOAD...**).

Для выполнения приложения, которое загружено, необходимо вызвать его в командной строке AutoCAD таким же образом, как вызывается любая функция AutoLISP.

Чтобы выполнить приложение **VLX**, необходимо загрузить Run-Time System (RTS) Visual LISP, а затем с помощью функции (**VL-LOAD...**) загрузить **VLX** модули. Например:

```
(VL-LOAD "C:/ALISP/KOTTEDJ.ARX")
```

Если приложение загружено, выполните его из командной строки AutoCAD так же, как и любую функцию AutoLISP.

Когда имя функции начинается с **C:**, ее можно вызывать, как команду AutoCAD, то есть заключать имя в круглые скобки не нужно.

Чтобы восстановить приложение после внесения в программу изменения, можно использовать Wizard (Мастер). Прежде чем восстанавливать приложение, необходимо удостовериться, что оно в настоящее время загружено в AutoCAD и разгрузить его. Visual LISP не будет производить запись поверх существующего модуля. Чтобы использовать Wizard (Мастер), выберите пункт **Existing Application Wizard** (Существующий мастер приложения) из всплывающего меню пункта **Make Application** (Создать приложение) падающего меню **File** (Файл) главного меню Visual LISP.

При создании приложения с помощью Wizard (Мастер) можно выбрать файл с расширением **МКР** и внести изменения в определение приложения.

Когда определение приложения не изменялось, приложение можно оперативно восстановить, если загрузить и выполнить файл с расширением MKP.

Если файл приложения MKP загружен в текстовом окне редактора Visual LISP, можно активизировать этот процесс, щелкнув по кнопке **Load active edit window** (Загрузить текст активного окна редактирования) на панели инструментов **Tools** (Инструменты). При загрузке файла с расширением MKP, Visual LISP выполняет команды в файле автоматически.

В качестве альтернативы Application Wizard (Мастер приложений) наиболее опытные (на современном языке – «продвинутые») пользователи могут использовать **Application API Visual LISP** для создания выполняемого модуля.

7.3. Встроенные функции Visual LISP

(VL-AUTOCAD-DEFUN) – определяет имя функции Visual LISP в AutoLISP;

(VL-AUTOCAD-UNDEFUN) – отменяет имя функции Visual LISP в AutoLISP. Т, если отменила, и NIL, если нет (например, функция не была определена в AutoLISP);

(VL-CONSP) – определяет, действительно ли переменная – список. Например:

```
_S (SETQ L '(A B C D))      ; нажать клавишу Ctrl
      (A B C D)
_$ (VL-CONSP L)             ; нажать клавишу Ctrl
Т
```

(VL-DIRECTORY-FILES) – ищет нужные файлы в каталоге (DIRECTORY); если каталог NIL или отсутствует, используется текущий каталог, если шаблон NIL или отсутствует, используется *.*; если число равно -1; ищутся только каталоги; если 0 – файлы и каталоги; если 1 – файлы. Например:

```
_S (VL-DIRECTORY-FILES      ; нажать клавиши Ctrl+Enter
"C:/PROGRAM FILES/AUTOCAD R14/VLISP" "*.ARX" 1)
(*Vlarts.arx" "Vlide.arx" "Vlrts.arx")
```

(VL-EVERY...) – проверяет, является ли условие истинным для каждой комбинации элементов списков (сначала первых элементов, затем вторых и т.д.). Например:

```
_S (SETQ L1 '(1 2 3 4))      ; нажать комбинацию клавиш Ctrl+Enter
      (SETQ L2 '(1 3 2 4))      ; нажать комбинацию клавиш Ctrl+Enter
      (VL-EVERY '< = L1 L2 )    ; нажать комбинацию клавиш Ctrl
```



```
(1 2 3 4)
```

```
(1 3 2 4)
```

```
nil
```

(VL-EXE-FILENAME) – возвращает полное имя файла текущей выполняемой программе. Например:

```
_$(VL-EXE-FILENAME)
```

```
"C:/PROGRAM FILES/AUTOCAD R14/VLISP/VLIDE.ARX"
```

(VL-FILE-COPY <"файл1"> <"файл2"> [число]) – копирует файл 1 и добавляет к файлу 2, если третий параметр есть и не NIL. Если возвращается NIL, то ошибка при копировании: файл 1 не читается, файл является каталогом, файл 2 существует. Например:

```
_$(VL-FILE-COPY "C:/AUTOEXEC.BAT" "C:/NEWAUTO.BAT")
```

```
1417
```

(VL-FILE-DELETE <"имя файла">) – удаляет файл. T, если успешно, и NIL, если нет;

(VL-FILE-DIRECTORY-P <"имя ">) – определяет, является ли имя именем каталога. T, если да, и NIL, если нет. Например:

```
_$(VL-FILE-DIRECTORY-P "PRIVETW")
```

```
NIL
```

(VL-FILE-RENAME <"новое имя"> <"старое имя">) – переименовывает файл. T, если переименовал, и NIL, если нет. Например:

```
(VL-FILE-RENAME "C:/PRIVETW.LSP" "C:/PRIVET1W.LSP")
```

```
T
```

(VL-FILE-SIZE <"имя файла">) – определяет размер файла в байтах. NIL, если имя файла нечитаемо, и 0, если имя файла – каталог или пустой файл:

```
_$(VL-FILE-SIZE "C:/AUTOEXEC.BAT")
```

```
533
```

(VL-FILE-SYSTIME <"имя файла">) – время последней модификации. Возвращает список, содержащий дату и время последнего изменения, или NIL, если файл не найден. Например:

```
_$(VL-FILE-SYSTIME "C:/ALISP/HANOIW.LSP ")
```

```
(1999 1 0 31 20 37 38 0)
```

Последний раз файл был изменен в 1999 году, в 1-м месяце года (январь), 0-й день недели (воскресенье), 31 января в 20:37:38;

(VL-FILENAME-BASE <"имя файла">) – возвращает имя файла после снятия пути DIRECTORY и расширения. Например:

```
_$(VL-FILENAME-BASE "C:/ALISP/HANOIW.LSP ")
```

```
"HANOIW"
```


(VL-FILENAME-DIRECTORY <"имя файла">) – возвращает путь каталога файла, без имени файла. Например:

```
_S (VL-FILENAME-DIRECTORY "C:/ALISP/HANOIW.LSP ")
"C:/ALISP"
```

(VL-FILENAME-EXTENSION <"имя файла">) – возвращает только расширение из имени файла, если его нет – NIL. Например:

```
(VL-FILENAME-EXTENSION "C:/PRIVETW.LSP")
".LSP"
```

(VL-FILENAME-MKTEMP [копируют расширение DIRECTORY]) – определяет уникальное имя файла, которое нужно использовать для временного файла. Основа имени – до 5 символов, принимаемых из имени файла, плюс 3 уникальных комбинации из символов. Все имена файла, сгенерированные функцией **(VL-FILENAME-MKTEMP...)** в течение сеанса Visual LISP удаляются, когда выходят из Visual LISP;

```
_S (VL-FILENAME-MKTEMP)
"C:\\WINDOWS\\TEMP\\$VL~~001"
```

(VL-INIT) – подражает инициализации AutoLISP на открытой или новой команде рисунка. Т, если инициализация прошла успешно;

(VL-LIST* <объект>...) – создает и возвращает список. Когда последний параметр – атом, результат – точечный список. Если последний параметр – список, то функция присоединяет ко всем предыдущим параметрам. Например:

```
_S (VL-LIST* 1)
1
_S (VL-LIST* 0 "НОЛЬ")
(0 . "НОЛЬ")
```

(VL-LIST-> STRING <список ASCII-кодов>) – преобразует список ASCII-кодов в элементы строки. Например:

```
_S (VL-LIST->STRING '(49 50 51 52))
"1234"
```

(VL-LIST-LENGTH <список>) – вычисляет длину списка. Например:

```
_S (VL-LIST-LENGTH '(A B C D E))
5
```

(VL-MEMBER-IF '<предикат> <список>) – определяет, является ли предикат (условие) истинным для одного из элементов списка. Предикат (условие) – это функция с одним аргументом, которая принимает значение Т для любого условия, назначенного пользователем.

Функция **(VL-MEMBER-IF..)** передает каждый элемент в списке функции, определенной в функции предиката. Если функциональные возвраты

не NIL, возвращается остальная часть списка тем же самым способом, как и в функции (**MEMBER...**);

```
_ $ (VL-MEMBER-IF 'LISTP '(1 "STR" (0. "STRING") NIL T))
((0.0 "STRING") nil T)
```

(**VL-MEMBER-IF-NOT** <предикат> <список>) – определяет, является ли предикат NIL для одного из элементов списка;

```
_ $ (VL-MEMBER-IF-NOT 'ATOM '(1 "STR" (0. "STRING") NIL T))
((0.0 "STRING") nil T)
```

(**VL-POSITION** <символ> <список>) – возвращает номер позиции определенного элемента списка, а первый элемент списка под номером 0, второй – 1 и так далее;

```
_ $ (SETQ STUFF (LIST "A" "B" "C" "D" "E"))
("A" "B" "C" "D" "E")
_ $ (VL-POSITION "D" STUFF)
3
```

(**VL-PRIN1-TO-STRING** <"объект">) – возвращает строковое представление любого объекта LISP, как будто это выводилось функцией (**PRIN1...**);

```
_ $ (VL-PRIN1-TO-STRING "BCDE")
"\BCDE\"
```

(**VL-PRINC-TO-STRING** <"объект">) – возвращает строковое представление любого объекта LISP, как будто это выводилось функцией (**PRINC**);

```
_ $ (VL-PRINC-TO-STRING "ABC")
"ABC"
```

(**VL-REGISTRY-WRITE** <"раздел реестра"> [<"имя"> <"значение">]) – создает в реестре Windows новый раздел, параметры, состоящие из имени и значения. Например:

```
_ $ (VL-REGISTRY-WRITE "HKEY_CURRENT_USER\\TEST"
"TEST NAME" "TEST DATA")
"TEST DATA"
```

Состояние реестра Windows после обращения к функции (**VL-REGISTRY-WRITE...**) показано на рис. 7.43;

(**VL-REGISTRY-READ** <раздел реестра> [<"имя"> <"значение">]) – чтение данных, сохраненных в реестре Windows;

```
_ $ (VL-REGISTRY-READ "HKEY_CURRENT_USER\\TEST" "TEST
NAME")
"TEST DATA"
```

(VL-REGISTRY-DELETE <"раздел реестра"> [<"имя"> <"значение">]) – удаляет из реестра Windows имя или значение;

Если регистр обеспечен и не NIL, заданная величина будет очищена от записи. Если значение имени отсутствует или NIL, функция удаляет определенную клавишу и все ее значения;

```
_S (VL-REGISTRY-DELETE "HKEY_CURRENT_USER\\TEST")
```

T

(VL-REGISTRY-DESCENDENTS <"раздел реестра"> [<список>]) – возвращает список имен подразделов выбранного раздела реестра;

```
_S (VL-REGISTRY-DESCENDENTS
```

```
"HKEY_LOCAL_MACHINE\\SOFTWARE")
```

```
("Mauisoft" "Microlog" "Quarterdeck" "Far" "DigitalEquipmentCorporation"
```

```
"SOC" "MachSoft" "Cthugha" "CyberLink" "Xing Technology Corp."
```

```
"Adobe" "Informatic" "Mouse Systems" "Kodak" "Ulead Systems" "Corel"
```

```
"Computer Artworks" "Symantec" "MicroQuill" "Autodesk" "ABBYY"
```

```
"PROJECT MT, Ltd." "Clients" "ODBC" "ACD Systems" "INTEL" "Description"
```

```
"CLASSES" "Microsoft")
```

(VL-REMOVE ' <элемент> <список>) – удаляет элемент из списка;

(VL-REMOVE PI (LIST PI T 0 "ABC"))

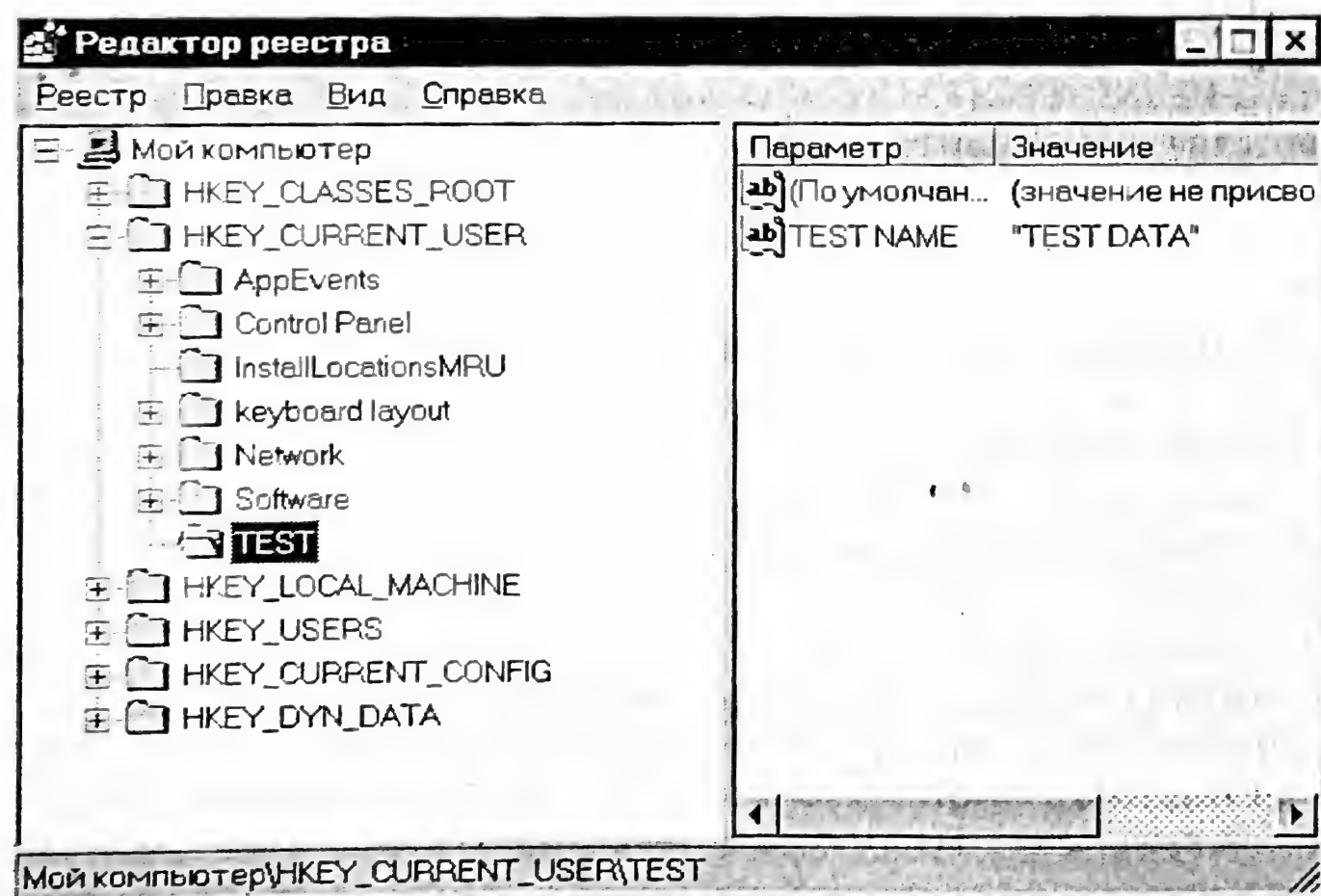


Рис. 7.43. Состояние окна редактора реестра Windows

(T 0 "ABC")

(VL-REMOVE-IF '<предикат> <список>) – удаляет все элементы списка, которые соответствуют условию;

```
_ $ (VL-REMOVE-IF 'LISTP (LIST '(PI T) 0 " ABC "))
(0 " ABC ")
```

(VL-REMOVE-IF-NOT '<предикат> <список>) – удаляет все элементы списка, которые не соответствуют условию;

```
_ $ (VL-REMOVE-IF-NOT 'LISTP (LIST '(PI T) 0 " ABC "))
((PI T))
```

(VL-SOME/ '<предикат> <список>...) – проверяет выполнение условия для первых элементов каждого списка, затем вторых и так далее;

```
_ $ (SETQ L1 (LIST 0 2 PI 4))
      L2 (CDR L1))
(VL-SOME '= L1 L2) ; сравнивает элементы списков
nil
```

(VL-SORT <список> '<функция сравнения>) – сортирует элементы в списке согласно функции сравнения. Функцией сравнения может быть любая функция, которая принимает два параметра и возвращает Т. Повторяющиеся элементы могут быть удалены из списка;

```
_ $ (VL-SORT '(3 2 1 3) '<); сортирует по возрастанию
(1 2 3)
```

(VL-SORT-I <список> '<функция сравнения>) – сортирует элементы в списке согласно функции сравнения и возвращает номера (индексы) элементов. Повторяющиеся элементы будут сохраняться;

```
_ $ (VL-SORT-I '("A" "D" "F" "C") '>); сортирует по убыванию
(2 1 3 0)
```

(VL-STRING->LIST <строка>) – преобразует элементы строки в список кодов ASCII;

```
_ $ (VL-STRING->LIST "132")
(49 51 50)
```

(VL-STRING-ELT <строка> <позиция>) – возвращает представление символа строки, стоящего в определенной позиции в строке в коде ASCII. Нулевой позиции соответствует первый символ в строке;

```
_ $ (VL-STRING-ELT "Успеха Вам, пользователи!!!" 5)
224 ; пояснение - на 5 позиции буква а, код ASCII ее - 224
```

(VL-STRING-LEFT-TRIM <строка удаляемых символов> <строка исходных символов>) – удаляет определенные символы с начала строки;

```
_ $ (VL-STRING-LEFT-TRIM "Пра" "Правнук")
"внук"
```

(VL-STRING-MISMATCH <строка1> <строка2> [<позиция1> <позиция2> <T>]) – возвращает длину совпадающего фрагмента символов для двух строк. Можно указать позиции начала фрагмента. Наличие буквы T снимает различие в написании строчных и прописных букв;

```
_S (VL-STRING-MISMATCH "Правнук" "Внук" 3 0 T)
```

```
4
```

(VL-STRING-POSITION <код символа> <строка> [<нач.поз.1> <T>]) – определяет позицию символа с определенным ASCII-кодом в строке. T – поиск с конца строки;

```
_S (VL-STRING-POSITION 224 "Читатель")
```

```
3 ; ASCII код буквы а - 224
```

(VL-STRING-RIGHT-TRIM <"удаляемые символы"> <строка>) – удаляет определенные символы с конца строки;

```
_S (VL-STRING-RIGHT-TRIM "як" "коньяк")
```

```
"конь"
```

(VL-STRING-SEARCH <"фрагмент"> <строка> [позиция начала поиска]) – определяет начальную позицию первого вхождения заданного фрагмента в строке. Поиск чувствителен к регистру;

```
_S (VL-STRING-SEARCH "Бар" "Санта-Барбара")
```

```
6
```

(VL-STRING-SUBST <"новый фрагмент"> <"старый фрагмент"> <исходная строка> [позиция начала поиска]) – заменяет старый фрагмент в исходной строке на новый. Поиск чувствителен к регистру;

```
_S (VL-STRING-SUBST "98" "95" "Windows 95")
```

```
"Windows 98"
```

(VL-STRING-TRANSLATE <"исходный набор"> <"новый набор"> <строка>) – заменяет исходный набор символов в строке на новый набор символов;

```
_S (VL-STRING-TRANSLATE "246" "357" "str2 str4 str6")
```

```
"str3 str5 str7"
```

(VL-STRING-TRIM <"удаляемый набор"> <строка>) – удаляет набор символов в начале и в конце строки;

```
_S (VL-STRING-TRIM "—" "—***—")
```

```
*****
```

(VL-SYMBOL-NAME <символ>) – возвращает строку, содержащую имя символа;

```
_S (VL-SYMBOL-NAME 'S::STARTUP)
```

```
"S::STARTUP"
```


(VL-SYMBOL-VALUE <символ>) – возвращает текущее значение, связанное с символом. Эта функция эквивалентна функции (EVAL...) AutoLISP, но не вызывает вычислитель LISP;

```
_$(VL-SYMBOL-VALUE 'T)
```

T

(VL-SYMBOLP <объект>) – идентифицирует, действительно ли определенный объект – символ;

```
_$(VL-SYMBOLP T)
```

T

(VLAX-3D-POINT <список>) – создает ACTIVEХ-совместимую трехмерную структуру точки. Список – 2 или 3 числа, представляет точку;

```
_$(VLAX-3D-POINT 5 20)
```

```
(5.0 20.0 0.0)
```

(VLAX-ADD-CMD <глобальное имя> <имя функции> [<локальное имя> <CMD-флажки>]) – добавляет команды к группе. Имя функции – функции AutoLISP с нулевыми параметрами;

CMD-флажки – целое число (значения по умолчанию к

ACRX_CMD_MODAL | ACRX_CMD_REDRAW);

(VLAX-CURVE-GETAREA <кривая>) – возвращает область внутри кривой;

(VLAX-CURVE-GETDISTATPARAM <кривая> <параметр>) – возвращает длину сегмента кривой от начала кривой до отметки, определяемой параметром;

(VLAX-CURVE-GETDISTATPOINT <кривая> <отметка>) – возвращает длину сегмента кривой между точкой начала кривой и определенной отметкой (в координатах WCS);

(VLAX-CURVE-GETENDPARAM <кривая>) – возвращает конечную точку кривой;

(VLAX-CURVE-GETENDPOINT <кривая>) – возвращает конечную точку кривой (в координатах WCS);

(VLAX-CURVE-GETPOINTATPARAM <кривая> <параметр>) – определяет точку на кривой, которая соответствует параметру, и возвращает эту точку;

(VLAX-CURVE-GETSTARTPARAM <кривая>) – возвращает параметр начала на кривой;

(VLAX-CURVE-GETSTARTPOINT <кривая>) – возвращает отметку начала кривой (в координатах WCS);

(VLAX-CURVE-ISCLOSED <кривая>) – определяет, закрыта ли определенная кривая;

(VLAX-CURVE-GETCLOSESTPOINTTO <кривая> <точка> [расширение]) – возвращает точку (в координатах WCS) на кривой, которая

является самой близкой к определенной точке. Если значение расширения не ноль, расширяет кривую при поиске самой близкой точки;

(VLAX-CURVE-GETCLOSESTPOINTTOPROJECTIONV <кривая> <точка> <нормаль> [расширение]) – возвращает точку (в координатах WCS) на кривой, которая является самой близкой к определенной точке. Если расширение не ноль, расширяет кривую при поиске самой близкой точки;

(VLAX-CURVE-GETFIRSTDERIV <кривая> <параметр>) – возвращает первую производную (в координатах WCS) кривой в определенном расположении;

(VLAX-CURVE-GETSECONDDERIV <кривая> <параметр>) – возвращает вторую производную (в координатах WCS) кривой в определенном расположении;

(VLAX-DUMP-OBJECT <ОБЪЕКТ VLA>) – перечисляет методы объекта и реквизиты;

(VLAX-ENAME->VLA-OBJECT <входное имя объекта>) – трансформирует объект в объект VLA (**VLISP APPLICATION**);

```
_S (SETQ E (CAR (ENTSEL)))
< ENTITY NAME 27E0540 >
_S (VLAX-ENAME->VLA-OBJECT E)
#<VLA-OBJECT IAUTOCADLWPOLYLINE 03F713A0 >
```

(VLAX-ERASED-P <объект VLA>) – определяет, был ли объект уничтожен. Т, если объект был уничтожен, иначе NIL;

(VLAX-FOR <символ> <набор объектов> [выражение1 [выражение2...]]) – выполняет итерации через набор объектов, оценивающих каждое выражение;

(VLAX-GET-AUTOCAD-OBJECT) – восстанавливает верхний уровень объекта приложения AutoCAD для текущего сеанса AutoCAD;

```
_S (SETQ AA (VLAX-GET-AUTOCAD-OBJECT))
#<VLA-OBJECT IAUTOCADApplication 00B3B91C>
```

(VLAX-INVOKE <объект VLA> <метод> < список параметров >) – вызывает определенный метод объекта. Функция (**VLAX-INVOKE...**) может использоваться для заказанного объекта **ACTIVE**;

```
(VLAX-INVOKE VLA-OBJECT "UPDATE" NIL)
; Это эквивалентно обращению к стандарту VLA:
(VLA-UPDATE VLA-OBJECT)
```

(VLAX-LDATA-DELETE <объект VLA или AutoCAD> <клавиша>) – стирает данные LISP из словаря рисунка. Т, если успешно, и NIL, если нет (например, данные не существовали);

```
_S (VLAX-LDATA-PUT "DICT" "KEY" '(1))
(1)
_S (VLAX-LDATA-DELETE "DICT" "KEY")
T
```

(VLAX-LDATA-GET <объект VLA или AutoCAD> <клавиша> [заданные по умолчанию данные]) – восстанавливает данные LISP из словаря рисунка;

```
_$ (VLAX-LDATA-GET "DICT" "KEY")
```

(1)

(VLAX-LDATA-LIST <объект VLA или AutoCAD>) – перечисляет данные LISP в словаре – ассоциативный список, состоящий из пар (клавиша – значение);

(VLAX-LDATA-PUT <объект VLA или AutoCAD> <клавиша> <сохраняемые данные>) – сохраняет данные LISP в словаре рисунка;

```
_$ (VLAX-LDATA-PUT "DICT" "KEY" '(1))
```

(1)

(VLAX-LDATA-TEST <данные LISP >) – определяет, могут ли данные быть сохранены за границей сеанса. Т, если данные могут быть сохранены и восстановлены за границей сеанса, NIL, если нет;

(VLAX-MAP-COLLECTION <объект VLA> <функция>) – применяет функцию ко всем объектам в наборе;

(VLAX-METHOD-APPLICABLE-P <объект VLA> <метод>) – определяет, поддерживает ли объект специфический метод;

```
_$ (VLAX-METHOD-APPLICABLE-P WHATSMYLINE "COPY")
```

T

(VLAX-OBJECT-RELEASED-P <объект VLA>) – определяет, был ли объект реализован;

(VLAX-PRODUCT-KEY) – возвращает путь записи AutoCAD. Этот путь может быть использован с целью регистрации приложения для загрузки запроса;

```
_$ (VLAX-PRODUCT-KEY)
```

```
"Software\\Autodesk\\AutoCAD\\R14.0\\ACAD-2451118:40800680"
```

(VLAX-PROPERTY-AVAILABLE-P <объект VLA> <свойство> [T]) – определяет, имеет ли объект определенное свойство

```
_$ (VLAX-PROPERTY-AVAILABLE-P MYCIRCLE "AREA")
```

T

Обратите внимание, как обеспечение факультативного третьего параметра изменяет результат:

```
_$ (VLAX-PROPERTY-AVAILABLE-P MYCIRCLE "AREA" T)
```

NIL

Функция возвращает NIL, хотя круг имеет свойство "AREA" («область»), которое не может изменяться;

(VLAX-PUT <объект VLA> <свойство> <значение>) – установка значения свойства функции нижнего уровня. Возвращает NIL, если установлено значение свойства;

```
_$(VLAX-PUT VLAOBJ "COLOR" 1)
NIL
```

(VLAX-READ-ENABLED-P <объект VLA>) – определяет, может ли объект читаться. Т, если объект читаем, и NIL, если нет;

(VLAX-REG-APP <имя регистрации приложения><список описаний команды> <INT> [имя приложения [булев флажок]]) – регистрирует приложение (**VLISP FUNCTION**).

Имя регистрации приложения – строка в форме "программное обеспечение\\<название компании>\\<название программы>\\<версия программы>".

Список описаний команды, каждое из которых должно быть одним из: (< GLOBAL-CMD-NAME >. < LOCAL-CMD-NAME >) или <CMD-NAME>. Каждое <CMD-имя> – строка или символ.

Булев флажок предотвращает сбой на операциях записи, если не NIL, и возвращает NIL; значение по умолчанию NIL.

Возврат – строка, содержащая путь раздела реестра, в котором было зарегистрировано приложение;

(VLAX-RELEASE-OBJECT <объект VLA>) – сбрасывает объект рисования. Возврат неопределенный;

(VLAX-REMOVE-CMD <глобальное имя>) – удаляет одиночную команду или целую группу команд для текущего сеанса AutoCAD;

(VLAX-TMATRIX <список>) – возвращает подходящее представление для матриц преобразования 4x4, которые нужно использовать в VLA методах. Список – список четырех списков, каждый из четырех чисел, представляет элементы матрицы преобразования;

(VLAX-TYPEINFO-AVAILABLE-P <объект VLA>) – определяет доступность **TYPELIB** информации. Т, если **TYPELIB** информация доступна, и NIL, если нет;

(VLAX-VLA-OBJECT-> ENAME <объект VLA>) – трансформирует объект VLA в объект AutoLISP. Возвращает имя объекта AutoLISP;

```
_$(VLAX-VLA-OBJECT->ENAME VLAOBJ)
<ENTITY NAME: 27E0540>
```

(VLAX-WRITE-ENABLED-P <объект VLA или входной объект AutoLISP>) – определяет, может ли объект рисования AutoCAD изменяться;

(VLISP-EXPORT-SYMBOL 'Symbol-name) – экспортирует местную переменную AutoLISP к значению, которое она имеет в Visual LISP;

```
_$(SETQ DD 1)
1
_$ (VLISP-EXPORT-SYMBOL 'DD)
1
```

(VLISP-IMPORT-EXSUBRS ("имя загрузочного модуля приложения" "названия функций приложения "...)) – регистрирует точку входа или приложения ARX внутри среды Visual LISP;

```
_$(VLISP-IMPORT-EXSUBRS '("AUTOCADAPP.EXE" "APPLoad"))
("AUTOCADAPP" "APPLOAD" "C:PSDRAG" "MTPROP" "MTEDIT" "C:PSFILL" "BHATCH"
"BPOLY" "C:PSIN" "AUTOCAD_COLORDLG" "STARTAPP" "ISMNUGRPLOADED"
"INITDIA")
```

(VLISP-IMPORT-SYMBOL <символ AutoLISP или список символов>) – назначает символ Visual LISP к тому же самому значению, которое он имеет в местном AutoLISP. Возвращает значение импортируемого символа или последнего импортируемого символа в списке символов;

```
_$(VLISP-IMPORT-SYMBOL 'DD)
```

1

(VLR-ACDB-REACTOR <данные AutoLISP> <список пар>) – создает базу данных объекта REACTOR (глобальной переменной). Список пар включает (<имя события>. <функцию повторного вызова>), где имя события – один из символов, перечисленных в таблице **ACDB**. Каждая функция повторного вызова принимает два параметра: REACTOR_OBJECT – объект VLR, называемый функцией повторного вызова, и OBJ – объект базы данных (передаваемый как объект AutoLISP), связанный с событием;

(VLR-ADD <объект VLR>) – доступ к заблокированному объекту REACTOR;

(VLR-ADDED-P <объект VLR>) – проверка допуска к объекту REACTOR. Т, если допускается, и NIL, если REACTOR заблокирован;

(VLR-BEEP-REACTION [параметры]) – выдает звуковой сигнал.

Эта функция работает только в Visual LISP IDE;

(VLR-CURRENT-REACTION-NAME) – возвращает имя текущего события, если вызывается изнутри повторного вызова объекта REACTOR;

(VLR-DATA-SET <объект VLR>) – возвращает специфические для приложения данные, связанные с объектом REACTOR;

(VLR-DATA-SET <объект VLR> <данные AutoLISP>) – записывает поверх специфические для приложения данные, связанные с объектом REACTOR;

(VLR-EDITOR-REACTOR <данные AutoLISP> <список пар>) – создает редактор объекта REACTOR. Объект REACTOR будет добавлен к базе данных рисунка, но не станет постоянным;

Список пар включает (<имя события> <функция повторного вызова>), где имя события – один из символов, перечисленных в «Таблице редактора». Каждая функция повторного вызова принимает два параметра: REACTOR_OBJECT – объект VLR, который называется функцией

повторного вызова, и OBJ – объект базы данных (переданный как объект AutoLISP), связанный с событием.

Список пар (< имя события > < список дополнительных элементов данных, связанных со специфическим событием>);

(VLR-LINKER-REACTOR <данные AutoLISP> < список пар>) – создает компоновщик объекта REACTOR. Список пар – (<имя события>, <функция повторного вызова >), где имя события – один из символов, перечисленных в «Таблице событий компоновщика». Каждая функция повторного вызова принимает два параметра: объект VLR и список, содержащий строку имени программы ARX;

(VLR-OBJECT-REACTOR <список AutoLISP объектов VLA > < данные AutoLISP> <список пар>) – создает объектный объект REACTOR. Список пар (<имя события> <функция повторного возврата>), где имя события – один из символов, перечисленных в «Таблице событий объекта». Каждая функция повторного вызова принимает три параметра: владелец объекта VLA, объект VLR и список дополнительных элементов данных;

(VLR-OWNER-ADD <объект VLR> <добавляемый объект VLR>) – добавляет объект к списку NOTIFIERS объектов VLR объектного REACTOR. Добавляет новый источник событий;

(VLR-OWNER-REMOVE <объект VLR> <удаляемый объект VLR>) – удаляет объект из списка NOTIFIERS объектного REACTOR;

(VLR-OWNERS <объект VLR>) – возвращает список объектов, которые сообщают определенному объекту REACTOR;

(VLR-PERS <объект VLR>) – делает объект REACTOR постоянным;

(VLR-PERS-P <объект VLR>) – определяет, действительно ли объект REACTOR постоянный;

(VLR-PERS-RELEASE <объект VLR>) – делает объект REACTOR нерезидентным;

(VLR-REACTION-NAMES <REACTOR-TYPE>) – возвращает список всех условий повторного вызова для этого типа объекта REACTOR (VLISP FUNCTION);

(VLR-REACTION-SET <объект VLR> <тип события> <функция>) – добавляет или заменяет функцию AutoLISP в объекте REACTOR;

Чтобы использовать функции, созданные в Visual LISP в AutoCAD, необходимо экспортировать имена функции в AutoCAD или ввести их неявно.

При явном экспортировании используют функцию (VL-AUTOCAD-DEFUN...);

(VL-AUTOCAD-DEFUN <'имя функции>) – экспортирует имя функции в AutoCAD;

По умолчанию любые функции Visual LISP, чьи имена начинаются с C., экспортируются в AutoCAD автоматически, если Visual LISP переменная системы *C-COLON-EXPORT* установлена как T.

ПРИЛОЖЕНИЕ

Графические примитивы AutoCAD и примеры их создания с использованием AutoLISP	346
Команды графического редактора AutoCAD для редактирования объектов ..	350
Представление данных в системе AutoCAD	352

В этой части книги приведены:

- *графические примитивы AutoCAD и примеры их создания с использованием в AutoLISP;*
- *команды графического редактора AutoCAD для редактирования объектов;*
- *представление данных в системе AutoCAD.*

Графические примитивы AutoCAD и примеры их создания с использованием AutoLISP

Наиболее часто используются следующие графические примитивы AutoCAD: точка, отрезок, окружность, дуга, полилиния, текст.

Графический примитив точка

Точка может быть задана несколькими способами:

1. Абсолютными координатами:

```
(COMMAND "POINT" "10, 12")
(SETQ T1 (GETPOINT "\n Введите координаты точки T1 ")) 10, 12
(COMMAND "POINT" T1)
(SETQ X (GETPOINT "\n Введите координату X точки T1 ") 10
      Y (GETPOINT "\n Введите координату Y точки T1 ") 12
(COMMAND "POINT" '(X Y))
(SETQ X 10.0
      Y 12.0)
(COMMAND "POINT" '(X Y))
(SETQ T1 '(X Y))
(COMMAND "POINT" T1)
(SETQ T1 (LIST X Y))
(COMMAND "POINT" T1)
(SETQ T1 (LIST 10.0 12.0))
(COMMAND "POINT" T1);
```

2. Относительными координатами:

```
(SETQ T1 (GETPOINT "\n Введите координаты точки T1 ")) 10,12
(SETQ T2 (GETPOINT "\n Введите координаты T2 относит. T1")) @5, 3;
```

3. Полярными координатами:

```
(SETQ U PI
      R 100
      T2 (POLAR T1 U R))
```

ИЛИ

```
(SETQ T2 (POLAR T1 PI 100));
```

4. Привязкой к характерным точкам примитива:

(OSNAP <исходная точка> <режим привязки>)

Основные способы привязки:

- (OSNAP T1 "NEA") – ближайшая (NEAREST) точка на примитиве;
- (OSNAP T1 "END") – конечная (ENDPOINT) точка линии или дуги;
- (OSNAP T1 "MID") – средняя (MIDDLE) точка линии или дуги;
- (OSNAP T1 "CEN") – центр (CENTER) дуги или окружности;

(OSNAP T1 "NOD") – ближайшая точка (вершина) (NODE);
(OSNAP T1 "QUA") – ближайшая точка, лежащая на оси квадранта по дуге и окружности (0, 90, 180, 270°);
(OSNAP T1 "INT") – точка пересечения (INTER-SECTION) двух примитивов (линий, дуг, окружностей);
(OSNAP T1 "INS") – точка вставки (INSERT) текста, блока;
(OSNAP T1 "PER") – точка опускания перпендикуляра на примитив;
(OSNAP T1 "TAN") – точка касания касательной к дуге или окружности;
(OSNAP T1 "NON") – отмена объектной привязки (NONE);
(OSNAP T1 "QUICK") – быстрый (QUICK) способ выбора объекта привязки.

Через запятую можно задать несколько режимов привязки. Приоритет действия привязки соответствует перечисленным выше.

Графический примитив отрезок

(COMMAND "LINE" T1 T2 "Ключ") – создание отрезка по двум точкам.

Ключи: "C" (Close) – замыкает ломаную;

"U" (Undo) – отменяет последний отрезок;

" " – завершает создание отрезка.

Например,

(COMMAND "LINE" T1 T2 T3 "C") – создание треугольника.

Графический примитив окружность

(COMMAND "CIRCLE" "Ключ" T1...) – создание окружности.

Если ключа нет – чертит окружность по центру и радиусу;

"2P" – чертит окружность по двум точкам на диаметре;

"3P" – по трем точкам на окружности;

"TTR" – (TANGENT TANGENT RADIUS) – чертит окружность, касающуюся двух примитивов и имеющую радиус R;

(COMMAND "CIRCLE" TC R) – по центру TC и радиусу R;

(COMMAND "CIRCLE" "2P" T1 T2) – по двум точкам T1 и T2 на диаметре;

(COMMAND "CIRCLE" "3P" T1 T2 T3) – по трем точкам на окружности;

(COMMAND "CIRCLE" "TTR" P1 P2 R) – обеспечивает касание двух примитивов P1 P2 и имеет радиус R.

Графический примитив дуга

(COMMAND "ARC"...) – создает дугу и имеет ключи:

"C" – (Centre) – центр дуги;

"L" – (Length) – длина хорды;

"S" – (Start) – начальная точка;
"R" – (Radius) – радиус дуги;
"E" – (End) – конечная точка;
"D" – (Direction) – направление;
"A" – (Angle) – центральный угол.

Примеры изображения дуги:

(COMMAND "ARC" T1 T2 T3) – изображение дуги по трем точкам;
(COMMAND "ARC" T1 "C" TC T2) – по двум точкам T1, T2 и центру TC;
(COMMAND "ARC" T1 "C" TC "A" U) – по точке, центру TC и углу U;
(COMMAND "ARC" T1 "C" TC "L" L) – по точке, центру TC и хорде L;
(COMMAND "ARC" T1 "E" T2 "R" R) – по точкам T1, T2 и радиусу R;
(COMMAND "ARC" T1 "E" T2 "U" U) – по двум точкам и центр.углу U;
(COMMAND "ARC" T1 "E" T2 "D" U) – по двум точкам и направлению U;
(COMMAND "ARC" "C" TC "S" T1 "E" T2) – по центру и двум точкам;
(COMMAND "ARC" "C" TC "S" T1 "A" U) – по центру, точке и углу U;
(COMMAND "ARC" "C" TC "S" T1 "L" L) – по центру, точке и хорде L;
(COMMAND "ARC" "CONTIN" T2) – продолжение линии или дуги в точку T2.

Графический примитив полилиния

(COMMAND "PLINE"...) – изображение последовательности прямолинейных и дуговых сегментов с возможностью указания ширины.

Ключи:

для измерения ширины:

"H" (Half-width) – полуширина;

"W" (Width) – ширина;

в режиме создания линий:

"A" (Arc) – переход в режим дуг;

"C" (Close) – замыкание линии;

"U" (Undo) – отмена последнего сегмента;

"L" (Length) – длина сегмента; ' '

в режиме создания дуг:

"A" (Angle) – центральный угол, градусы;

"CE" (CEnter) – центр дуги;

"CL" (CLose) – замыкание дугой;

"D" (Direction) – направление по касательной;

"L" (Line) – переход в режим линий;

"L" (Length) – длина хорды;

"R" (Radius) – радиус дуги;

"S" (Second P.) – вторая точка.

Например,

```
(COMMAND "PLINE" T1 "W" 0.5 0.5 T2 T3  
"A" T4 "LINE" T5 T6 "A" T7 " ").
```

Графический примитив эллипс

(COMMAND "ELLIPSE...") – построение эллипса с ключами:

"C" (Center) – центр эллипса;

"R" (Rotation) – угол поворота;

(COMMAND "ELLIPSE" T1 T2 D2) – по двум точкам на концах главной оси T1, T2 и длине другой оси – D2;

(COMMAND "ELLIPSE" T1 T2 "R" U) – по двум точкам на концах главной оси и углу поворота относительно оси;

(COMMAND "ELLIPSE" "C" TC T1 D2) – по центру TC, точке на конце главной оси и величине другой оси эллипса;

(COMMAND "ELLIPSE" "C" TC T1 "R" U) – по центру, точке на конце главной оси и углу поворота относительно оси.

Графический примитив кольцо

(COMMAND "DONUT" D1 D2 TC) – изображение кольца по внутреннему диаметру кольца D1, внешнему – D2 и центру – TC.

Графический примитив многоугольник

(COMMAND "POLYGON"...) – построение прямоугольника с ключами: "E" (EDGE) – сторона; "C" – описывающий многоугольник; "I" – вписанный многоугольник;

(COMMAND "POLYGON" N "E" T1 T2) – по числу сторон N и двум точкам, лежащим на концах стороны;

(COMMAND "POLYGON" N "C" TC "I" R) – по числу сторон N, центру TC и радиусу вписанной окружности R;

(COMMAND "POLYGON" N "C" TC "C" R) – по числу сторон N, центру TC и радиусу описывающей окружности R.

Графический примитив текст

(COMMAND "TEXT"...) – представление текста на чертеже;

(COMMAND "TEXT" "A" T1 T2 "...") – вписывание текста между двумя точками T1 и T2;

(COMMAND "TEXT" "F" T1 T2 "H" H "...") – ввод текста по двум точкам и высоте H;

(COMMAND "TEXT" "C" TC "R" U "H" H "...") – ввод текста по цент-

ральной точке (Center) TC, углу поворота U и высоте букв H;

(COMMAND "TEXT" "M" TC "R" U "H" H "...") – ввод текста по середине (MIDDLE) с углом поворота U и высоте шрифта H;

(COMMAND "TEXT" "R" T2 "R" U "H" H "...") – ввод текста с выравниванием по правому (RIGHT) краю T2, углу поворота U и высоте шрифта H;

(COMMAND "TEXT" T1 "R" U "H" H "...") – ввод текста по начальной точке T1, углу поворота U и высоте шрифта H;

(COMMAND "DTEXT"...) – ввод нескольких строк текста. Ключи и способы ввода текста аналогичны команде "TEXT".

Команды графического редактора AutoCAD для редактирования объектов

Объект в графическом редакторе AutoCAD – это некоторый примитив или совокупность примитивов (набор). Графический редактор AutoCAD обеспечивает возможность редактирования объектов. Первым шагом перед редактированием является выбор объектов, который может быть произведен с помощью функции присвоения (SETQ...) языка AutoLISP или с помощью специальных ключей редактора AutoCAD:

"A" (Addition – прибавление) – добавление к набору еще одного;

"B" (BOX) – выбор объектов рамкой с угловыми точками T1 и T2.

Если T2 справа от T1, то выбор объектов осуществляется внутри рамки, если T2 слева от T1, то выбираются объекты, как находящиеся внутри рамки (окна), так и пересекающие ее "B" N1 N2 "";

"C" (Crossing – пересечение) – выбор объектов, находящихся как внутри рамки с угловыми точками T1 и T2 (окна), так и пересекающихся ею "C" T1 T2 "";

"L" (Last – последний) – выбор последнего изображенного объекта;

"M" (Multiple – многократно) – многократный выбор объектов;

"P" (Point – точка) – выбор объектов, на которых лежит точка.

Если точка не лежит на объекте, то она становится первым углом рамки (окна), "P" T1 "";

"PR" (Previous – предыдущий) – выбор предыдущего набора объектов;

"S" (Single – единственный) – выбор одного объекта;

"R" (Remove – удаление) – удаление указанного объекта из набора;

"U" (Undo – отмена) – удаление последнего добавленного в набор объекта;

"W" (Window – окно) – выбор объектов, находящихся внутри рамки (окна) с угловыми точками T1 и T2.

В графическом редакторе AutoCAD имеется целый набор команд для

редактирования выделенных объектов:

(COMMAND "MOVE" <выбор объектов> BT1 BT2 "") – перемещение выбранных объектов с базовой точкой BT1 к BT2;

(COMMAND "ERASE" <выбор объектов> "") – удаление объектов;

(COMMAND "COPY" <выбор объектов> BT1 BT2 "") – копирование выбранных объектов с базовой точкой BT1 и размещение в BT2;

(COMMAND "ROTATE" <выбор объектов> TC U "") – поворот выбранных объектов относительно центра TC на угол U против часовой стрелки;

(COMMAND "SCALE" <выбор объектов> TC K "") – изменение масштаба относительно центра TC с коэффициентом масштаба K;

(COMMAND "MIRROR" <выбор объектов> T1 T2 "") – зеркальное отражение выбранных объектов относительно линии, проходящей через точки T1 и T2;

(COMMAND "OFFSET" "T" <выбор объектов> T1 "") – изображение подобного объекта, проходящего через точку T1;

(COMMAND "ARRAY" <выбор объектов> "R" NR NS RR RS "") – создание прямоугольного массива примитивов (объектов) с числом рядов NR, числом столбцов NS и расстояниями между рядами RR и столбцами RS;

(COMMAND "ARRAY" <выбор объектов> "P" TC N U "Y" "") – формирование кругового массива (по кругу) с центром TC, числом объектов N и размещением их против часовой стрелки с углом перемещения U и возможностью ориентации объекта Y (N);

(COMMAND "STRETCH" <выбор объектов> BT1 BT2 "") – вытягивание выбранных объектов с базовой точкой BT1 и сдвиг их в BT2;

(COMMAND "DIVIDE" <выбор объектов> N "") – деление объекта на N частей;

(COMMAND "DIVIDE" <выбор объектов> "B" NB "Y" N "") – укладка на выделенный объект блока NB и возможной ориентацией (YES) заданное число раз N;

(COMMAND " " <выбор объекта> "B" NB1 "Y" R "") – измерение объекта с помощью блока NB1 с расстоянием между блоками R и возможностью ориентации (YES) (NOT);

(COMMAND "MEASURE" <выбор объекта> T1 T2 "") – измерение объекта единицей измерения – отрезок [T1 T2];

(COMMAND "CHANGE" <выбор объекта> T1 U "") – изменения: для отрезка – его конечная точка передвигается в точку T1; для окружности – меняет радиус; для текста и блока – меняется точка вставки; примитив поворачивается на угол U;

(COMMAND "CHANGE" <выбор объекта> "P"... "") – изменение свойств примитива: "C" (COLOR – цвет); "LA" (LAYER – слой); "LT" (LTYPE – тип линии). За ключом "C" стоит код нового цвета, за ключом

"LA" – новое имя слоя; за "LT" – новый тип линии ("DASHDOT" – штрих-пунктирная, "DASHED" – пунктирная, "CONTINUOUS" – непрерывная).

(COMMAND "BREAK" <выбор объекта> T1 T2 "") – разрыв без стирания или стирание части объекта от точки T1 ("F") до точки T2 против часовой стрелки. Если указан ключ "F", разрыв в T1;

(COMMAND "TRIM" <выбор границ> "" <выбор объекта> "") – удаление части примитивов, которые пересекают границы;

(COMMAND "EXTEND" <выбор границы> "" <выбор объекта> "") – вытягивание объекта до границы;

(COMMAND "FILLET" "R" R <выбор объектов> "") – сопряжение линий: если стоит ключ "R", а за ним значение R, сопряжение производится по радиусу R двух линий, которые выбираются после данной команды; если стоит ключ "P", скругляется полилиния, которая выбирается сразу же за ключом.

(COMMAND "CHAMBER" "D" D1 D2 <выбор двух линий> "") – вычерчивание фасок: если стоит ключ "D" и указаны расстояния отсечения линий от точек их пересечения – D1 и D2, то концы отсеченных линий соединяются; если стоит ключ "P", в углах полилинии снимаются фаски;

(COMMAND "PEDIT" <выбор полилинии> "Ключ" "") – редактирование полилиний с использованием соответствующих ключей:

"C" (CLOSE – закрыть) – замыкание ломаной линии;

"O" (Open – открыть) – размыкание ломаной линии;

"J" (Join – присоединить) – присоединение дуг и ломаных линий;

"W" (Width – ширина) – ввод новой ширины ломаной линии;

"E" (Edit – редактировать) – выбор и редактирование вершин;

"F" (Fit – скруглить) – скругление вершин ломаной линии;

"S" (Spline – сплайн) – аппроксимация ломаной линии сплайном;

"D" (Decurve – выпрямить) – удаление скруглений;

"U" (Undo – отменить) – отмена режима редактирования;

"X" (Exit – выйти) – выход из режима редактирования.

Например, (COMMAND "PEDIT" <выбор ломаной> "F" "X").

Представление данных в системе AutoCAD

AutoCAD – одна из универсальных базовых систем автоматизированного конструирования, обеспечивающая возможность эффективного выполнения операций конструирования. Основным элементом этой системы является примитив. Примитив – это элемент чертежа, воспринимаемый системой как нечто целое. Примитивы в AutoCAD могут быть простыми (отрезок, окружность, дуга и т. д.) и составными (полилиния, вставки, блоки).

Составные примитивы делятся на субпримитивы. Например, для полилинии субпримитивами являются вершины, для вставки, блоков – атрибуты.

Вся информация о примитивах в системе AutoCAD содержится в базе данных графического редактора AutoCAD в виде списков. Список состоит из подсписков, большинство которых имеют вид точечных пар. Исключение составляют подписки, описывающие координаты точки. Первый элемент подписка – это код (ключ) параметра, а второй и последующие элементы – значения параметра. Каждый примитив характеризуется совокупностью параметров: именем, типом, именем типа линии, слоя и т. д. Код и значение параметра в точечной паре разделяются точкой. В таблице приведены примеры, коды и параметры для примитивов, которые используются наиболее часто.

<i>Код и значение параметра</i>	<i>Отрезок</i>	<i>Окружность</i>	<i>Дуга</i>
(-1.<имя примитива>)	(-1. L4)	(-1. CR2)	(-1. AR)
(0.<тип примитива>)	(0."LINE")	(0."CIRCLE")	(0."ARC")
(6.<имя типа линии>)	(6."DASHED")	(6."CEnter")	(6."DASHED")
(8.<имя слоя>)	(8. 0)	(8. "SLOI1")	(8. "SLOI\$")
(10.<коорд. точки>)	(10 3 4)	(10 50 50)	(10 77 15)
(11.<коорд. точки>)	(11 9 8)		
(40.<радиус>)		(40. 150)	(40. 225)
(50.<начальный угол>)			(50. 30)
(51.<конечный угол>)			(51. 77)
(62.<цвет>)	(62. 5)	(62. 3)	(62. 4)

Чтобы вывести список (совокупность подсписков) данных любого примитива, необходимо использовать функцию (**ENTGET** <имя примитива>). Например, список данных примитива с именем L4, а под этим именем ранее был создан отрезок, можно получить следующим образом:

```
(ENTGET L4)
((-1.6000053C) (0 . "LINE") (6 . "DASHDOT") (8 . "SL")
(10 5 70) (11 55 90) (62 . 5))
```

Из списка данных следует, что под этим именем отрезок. По существу, список данных примитива – это ассоциативный список, список типа ("код параметра" – "значение параметра"). Код (ключ) определяет первый элемент подписка, а второй элемент подписка определяет значение параметра.

Используя функцию (**ASSOC** <код (ключ) параметра> <список>), можно выделить нужный подсписок. Чтобы определить, например, какой примитив скрывается под именем L4, достаточно использовать функцию

(ASSOC 0 L4)

(0 . "LINE")

Определить координаты центра окружности CR2 (см. таблицу) можно, например, используя несколько функций:

(SETQ TC (CDR (ASSOC 10 CR2)))

Функция (ASSOC 10 CR2) определяет подсписок вида (10 50 50), функция (CDR...) формирует список без первого элемента (50 50), функция (SETQ...) присваивает переменной TC координаты центра окружности (50 50).

В качестве графического примитива в AutoCAD может выступать блок. Блок может содержать любое количество графических примитивов любого типа, а восприниматься AutoCAD как один графический примитив. Для создания блока (только в текущем чертеже) используется команда "BLOCK". Для создания блоков, которыми можно воспользоваться при построении любых чертежей, достаточно выбрать команду "WBLOCK". Эта команда служит для записи блока в отдельный файл с заданным именем и имеет расширение DWG. Рассмотрим примеры создания блока из выбранных объектов:

(COMMAND "BLOCK" <имя блока> <базовая точка блока> <выбор объектов> "")

(COMMAND "BLOCK" BL5 BT5 L4 CR2 AR "") .

Запись блока в файл:

(COMMAND "WBLOCK" F1 BL5) .

Включение блока в текущий чертеж производится при помощи команды "INSERT" (Вставить), которая в общем виде выглядит так:

(COMMAND "INSERT" <имя блока> <базовая точка вставки> <масштаб по оси X> <масштаб по оси Y> <угол поворота>). Например,

(COMMAND "INSERT" BL5 BT1 1 1 U) .

Графический редактор AutoCAD позволяет создавать чертежи, которые представляют собой «многослойный пирог». Например, на первом слое дается чертеж без размеров, на втором – только размеры к данному чертежу, на третьем – штриховка к отдельным его элементам. При необходимости слои можно накладывать друг на друга в любом сочетании, удалив ненужные.

Для реализации многослойного изображения чертежа используется команда "LAYER", которая имеет следующие ключи:

"M" (Make – сделай) – создает слой и делает его текущим;

"S" (Set – установи) – делает слой текущим;

"ON" (ON – включи) – делает слой видимым;

"OFF" (OFF – отключи) – делает слой невидимым;

"C" (Color – цвет) – определяет цвет слоя;

"N" (New – новый) – создает новый слой;

"L" (Ltype – тип линии) – определяет тип линии;

"?" – выводит список созданных слоев.

Ключ "C" (Color – цвет) может использовать следующие цвета: "R" (Red – красный), "Y" (Yellow – желтый), "G" (Green – зеленый), "C" (Cyanic – голубой), "B" (Blue – синий), "M" (Magenta – розовый), "W" (White – белый).

Ключ "L" (Linetype – тип линии) может использовать следующие режимы:

"S" (Set – установи) – текущий тип линии: "CONTINUOUS" – непрерывный; "DASHED" – пунктирный; "DASHDOT" – штрихпунктирный;

"R" (Read – просмотр) – просмотр типов линий;

"C" (Create – создание) – создание нового типа линий;

"L" (Load – загрузка) – использование нового типа линий.

Например, (COMMAND "LAYER" "M" "L" "DASHDOT" "").

При использовании команд построения графических примитивов часто применяют команды: "ZOOM"; "UCS".

"ZOOM" (Покажи) – изменяет размер, а также местоположение чертежа на экране и имеет следующие ключи:

"C" (Center – центр) – изображение чертежа по центру;

"D" (Dynamic – динамический) – изображение динамическим окном;

"L" (Left – левый) – задание левого нижнего угла чертежа с изменением масштаба;

"P" (Previous – предыдущий) – предыдущий режим изображения;

"W" (Window – окно) – изображение окном с угловыми точками.

Например,

(COMMAND "ZOOM" "L" T1 M "") .

"UCS" (User Coordinate System) – система координат пользователя, которая имеет следующие ключи:

"O" (Origin – начало) – параллельный перенос координат;

"Z" (Z – ось) – поворот вокруг оси Z на заданный угол;

"S" (Save – сохрани) – сохранение текущей системы координат под некоторым именем.

Например, (COMMAND "UCS" "S" IM1).

"R" (Restore – восстанови) – восстановление под заданным именем;

"W" (World – мир) – переход в абсолютную систему "WCS" – World Coordinate System.

АЛФАВИТНЫЙ УКАЗАТЕЛЬ

А

Аргументы /определение/ 47

Б

База знаний 231

В

Выражение /определение/ 46

И

Идентификаторы /определение/ 47

Интерфейс 231

Инфиксная форма /определение/ 52

К

Кнопка управления 17

Константы /определение/ 47

Л

Логические константы /определение/ 47

Логическое исчисление 232

М

Механизм вывода 231

О

Объект 350

Окно консоли /определение/ 40

П

Переключатель 17
Переменные /определение/ 47
Предикат 232
Префиксная форма 51
Примитив 352
Производственные системы 232
Прокручиваемый список 18

Р

Раскрывающийся список 18

С

Символы
1+ /функция AutoLISP/ 62
1- /функция AutoLISP/ 62
Семантические сети 232
Символ /определение/ 47
Список /определение/ 46

Т

Текстовые константы /определение/ 47

Ф

Флажок 17

Ч

Числовые константы /определение/ 47

Э

Экспертная система 231

А

ABS /функция AutoLISP/ 62
ACTION_TILE /функция AutoLISP/ 70
Activate AutoCAD /команда Visual LISP/ 31
Add Watch /команда Visual LISP/ 28

ADD_LIST /функция AutoLISP/ 70
ADS /функция AutoLISP/ 71
AGR /функция пользователя/ 246
ALLOC /функция AutoLISP/ 71
AND /функция AutoLISP/ 65
ANGLE /функция AutoLISP/ 63
Animate /команда Visual LISP/ 29
APPEND 77
APPLY /функция AutoLISP/ 64, 177
Apropos Window /команда Visual LISP/ 82
Arrange Icons /команда Visual LISP/ 31
ARX /функция AutoLISP/ 71
ARXUNLOAD /функция AutoLISP/ 71
ASCII /функция AutoLISP/ 60
ASSOC 60
ATAN /функция AutoLISP/ 62
ATOF /функция AutoLISP/ 60
ATOM /функция AutoLISP/ 65
ATOMS-FAMILY /функция AutoLISP/ 71
AUTOARXLOAD /функция AutoLISP/ 71
AutoCAD /определение/ 9
AutoCAD Mode /команда Visual LISP/ 29, 30, 82
AUTOCAD_HELPDLG HELPFILE TOPIC /функция AutoLISP/ 70
AUTOCAD_STRLSORT /функция AutoLISP/ 70
AutoLISP /определение/ 46
AUTOLOAD /функция AutoLISP/ 71
AUTOXLOAD /функция AutoLISP/ 71

В

BLIPMODE 70
BookMarks /команда Visual LISP/ 27
BOUNDP /функция AutoLISP/ 65
BOX /функция AutoLISP/ 145
BOX /функция пользователя/ 108
Browse Drawing Database /команда Visual LISP/

С

С /функция пользователя/ 202
CAAR /функция AutoLISP/ 59
CADAR /функция AutoLISP/ 59

CADR /функция AutoLISP/ 59
CANS /функция пользователя/ 293
Cascade /команда Visual LISP/ 31
CHR /функция AutoLISP/ 60
Clear Console Window /команда Visual LISP/ 25
Close Windows /команда Visual LISP/ 31
CMDECHO 70
COMMAND /функция AutoLISP/ 72
Complete Word by Match /команда Visual LISP/ 26
COND /функция AutoLISP/ 63
Configure Current /команда Visual LISP/ 30
Console History Down /команда Visual LISP/ 26
Console History UP /команда Visual LISP/ 26
COPYL /функция пользователя/ 74
COS /функция AutoLISP/ 62
CP /функция пользователя/ 191
CVUNIT /функция AutoLISP/ 73

D

DEFUN /функция AutoLISP/ 69
DEL /функция пользователя/ 293
Delete /команда Visual LISP/ 25
DELOP /функция пользователя/ 167
DICTADD /функция AutoLISP/ 72
DICTNEXT /функция AutoLISP/ 72
DICTREMOVE /функция AutoLISP/ 72
DICTRENAME /функция AutoLISP/ 72
DIMASZ /команда AutoCAD/ 133
DIMDLI /команда AutoCAD/ 133
DIMEXE /команда AutoCAD/ 133
DIMTSZ /команда AutoCAD/ 133
DIMY_TILE /функция AutoLISP/ 72
DISTANCE /функция AutoLISP/ 63
DONE_DIALOG /функция AutoLISP/ 72
DVIEW /команда AutoCAD/ 146

E

EARLYAB 74
EE /функция пользователя/ 224
EKSPERT /функция пользователя/ 236

ELKTRISL /функция пользователя/ 198
END_IMAGE /функция AutoLISP/ 72
END_LIST /функция AutoLISP/ 72
ENTLAST /функция AutoLISP/ 67
ENTNEXT /функция AutoLISP/ 67
ENTSEL /функция AutoLISP/ 68
Environment Options /команда Visual LISP/ 30
EQ /функция AutoLISP/ 66
EQUAL /функция AutoLISP/ 65
Error Trace /команда Visual LISP/ 27
EVAL /функция AutoLISP/ 69
EVALD /функция пользователя/ 166
EVALV /функция пользователя/ 167
Existing Application WIZARD /команда Visual LISP/ 23
EXIT /команда Visual LISP/ 24
EXP 62
EXPT /функция AutoLISP/ 62

F

FINDFILE /функция AutoLISP/ 70
FIX /функция AutoLISP/ 60
FLOAT /функция AutoLISP/ 60
Font /команда Visual LISP/ 30
FOREACH /функция AutoLISP/ 64

G

GETCFG /функция AutoLISP/ 73
GETCORNER /функция AutoLISP/ 57
GETDIST /функция AutoLISP/ 57
GETINT /функция AutoLISP/ 57
GETPOINT /функция AutoLISP/ 57
GETREAL /функция AutoLISP/ 57
GETSTRING /функция AutoLISP/ 57
GP+ /функция пользователя/ 191
GP- /функция пользователя/ 191

H

HANDENT /функция AutoLISP/ 68

I

Iconize All /команда Visual LISP/ 31
INFPREF /функция пользователя/ 162
Inspect /команда Visual LISP/ 27
ITOA /функция AutoLISP/ 61

K

KLETKA /функция пользователя/ 291
KO /функция пользователя/ 202

L

L /функция пользователя/ 202
LAMBDA /функция AutoLISP/ 69
LAST /функция AutoLISP/ 59
LAYER /команда AutoCAD/ 354
LENGTH /функция AutoLISP/ 61
LINE /команда AutoCAD/ 347
LISP 9
LISP Console /команда Visual LISP/ 28
LISTP /функция AutoLISP/ 65
LOAD /функция AutoLISP/ 70
Load File /команда Visual LISP/ 23
LOG /функция AutoLISP/ 62
LP /функция пользователя/ 191

M

Make Application /команда Visual LISP/ 23
MAPCAR /функция AutoLISP/ 64
Match Backward /команда Visual LISP/ 26
Match Forward /команда Visual LISP/ 25
MAX /функция AutoLISP/ 62
MEASURE /команда AutoCAD/ 351
MEMBER /функция AutoLISP/ 60
MIN /функция AutoLISP/ 63
MINUSP /функция AutoLISP/ 65
MODER /функция пользователя/ 79
MODES /функция пользователя/ 79

N

New Application WIZARD /команда Visual LISP/ 23

New File /команда Visual LISP/ 19

New Project /команда Visual LISP/ 28

NOT /функция AutoLISP/ 65

NTH /функция AutoLISP/ 60

NUMBERP /функция AutoLISP/ 65

O

OFFICE /функция пользователя/ 111

Open File /команда Visual LISP/ 20

Open Project /команда Visual LISP/ 28

OR /функция AutoLISP/ 65

Organize /команда Visual LISP/ 31

P

PAR /функция пользователя/ 203

PAR1 /функция пользователя/ 284

Parentheses Matching /команда Visual LISP/ 25

PARN /функция пользователя/ 203

Paste /команда Visual LISP/ 24, 25, 26

PERMN /функция пользователя/ 236

PODMNOJ /функция пользователя/ 236

POLAR /функция AutoLISP/ 63

POSL /функция пользователя/ 202

POSL1 /функция пользователя/ 281

POSLN /функция пользователя/ 202

PPAR1 /функция пользователя/ 282

PREINF /функция пользователя/ 171

PRIN1 /функция AutoLISP/ 66

PRINC /функция AutoLISP/ 66

PRINT /команда Visual LISP/ 21

PRINT /функция AutoLISP/ 66

Print Setup /команда Visual LISP/ 23

PROGN /функция AutoLISP/ 64

PROMPT /функция AutoLISP/ 57

Q

QUOTE /функция AutoLISP/ 69

R

R /функция пользователя/ 202
READD /функция пользователя/ 235
READF /функция пользователя/ 234
READL /функция пользователя/ 237
READP /функция пользователя/ 235
Redo /команда Visual LISP/ 82
REDRAW /функция AutoLISP/ 73
REM /функция AutoLISP/ 63
Reopen /команда Visual LISP/ 20
REVERSE /функция AutoLISP/ 59
REZONANS /функция пользователя/ 203
RP /функция пользователя/ 190
RTOS /функция AutoLISP/ 61

S

Save ALL /команда Visual LISP/ 21
Save Setting /команда Visual LISP/ 30
Select Backward /команда Visual LISP/ 26
Select Forward /команда Visual LISP/ 26
SET /функция AutoLISP/ 59
SETQ /функция AutoLISP/ 59
SETR 132
SETR /функция пользователя/ 128
SETRIS /функция пользователя/ 216
SETVAR /функция AutoLISP/ 70
SFERMA1 /функция пользователя/ 264
SHEMARIS /функция пользователя/ 197
SIN /функция AutoLISP/ 62
SOBN /функция пользователя/ 215
SSADD /функция AutoLISP/ 69
SSDEL /функция AutoLISP/ 69
SSGET /функция AutoLISP/ 68
SSLENGTH /функция AutoLISP/ 69
SSNAME /функция AutoLISP/ 69
Stop Once /команда Visual LISP/ 28
STRCAT /функция AutoLISP/ 61
STRLEN /функция AutoLISP/ 61
STUP /функция пользователя/ 124
SUBST /функция AutoLISP/ 60

SUBSTR /функция AutoLISP/ 61
Symbol Service /команда Visual LISP/ 27

T

TEXTBOX /функция AutoLISP/ 68
TEXTIN1 /функция пользователя/ 134
TEXTPAGE /функция AutoLISP/ 73
TEXTSCR /функция AutoLISP/ 73
Tile Horizontally /команда Visual LISP/ 30
Tile Vertically /команда Visual LISP/ 30
Toggle Console Log /команда Visual LISP/ 23, 82
Trace Command /команда Visual LISP/ 28
Trace Stack /команда Visual LISP/ 27

U

U /функция пользователя/ 166
U* /функция пользователя/ 164
U+ /функция пользователя/ 163
U- /функция пользователя/ 163
U/ /функция пользователя/ 165
U= /функция пользователя/ 166
U^ /функция пользователя/ 166
UNTRACE /функция AutoLISP/ 70

V

Visual LISP /определение/ 12
VL-FILE-DIRECTORY-P 331
VL-FILENAME-EXTENSION /функция Visual LISP/ 332
VL-REMOVE /функция Visual LISP/ 334
VL-SOME /функция Visual LISP/ 335
VLAX-CURVE-GETAREA /функция Visual LISP/ 337
VLAX-CURVE-GETDISTATPOINT /функция Visual LISP/ 337
VLAX-PROPERTY-AVAILABLE-P /функция Visual LISP/ 339
VLAX-WRITE-ENABLED-P /функция Visual LISP/ 340
VLR-BEEP-REACTION /функция Visual LISP/ 341
VP+ /функция пользователя/ 191
VP- /функция пользователя/ 191

W

Watch Last Evaluation /команда Visual LISP/ 28

Watch Window /команда Visual LISP/ 27

WEDGE /функция AutoLISP/ 145

WHILE /функция AutoLISP/ 64

Window Attributes /команда Visual LISP/ 30

WRITE-CHAR /функция AutoLISP/ 66

WRITE-LINE /функция AutoLISP/ 66

Z

Zoom /команда Visual LISP/ 31



AutoCAD 14

Русская и англоязычная версии

Автор: Романычева Э.Т. и др.
Формат: 70×100 1/16
Страниц: 512
ISBN: 5-89818-004-4

Книга является практическим руководством, основой для самостоятельного изучения и освоения мощной универсальной среды автоматизации инженерно-графических работ AutoCAD. Она поможет читателю в короткий срок освоить команды двухмерного черчения, трехмерного моделирования и получения конструкторской документации по разработанным проектам.

Кроме того, в книге предлагаются различные методики, подходы к разработке конструкторской документации, проверенные многолетним опытом работы в среде AutoCAD, которые могут быть приняты целиком для использования либо взяты за основу независимо от версии AutoCAD.

Книга сопровождается большим количеством иллюстраций, реализованных в системе AutoCAD, и справочными данными; содержит упражнения, охватывающие практически все команды и иллюстрирующие их возможности. К книге прилагается электронный вариант упражнений на компакт-диске.

Очень важно, что благодаря максимальной преемственности по командам и по структуре данных, навыки, приобретенные при выполнении данных в книге упражнений и их электронного варианта, можно использовать в среде AutoCAD различных версий. Иллюстрации рабочего стола AutoCAD R14 выполнены в Windows NT.

Книга предназначена для широкого круга пользователей: учащихся любых учебных заведений, в том числе студентов вузов, профессиональных чертежников, конструкторов и инженеров, которые хотят освоить и использовать современные компьютерные технологии в конструировании и проектировании.

Для легальных пользователей.

ДЛЯ СВЯЗИ:

Оптовые закупки: тел. (095) 264-7536
E-mail: info@dmk.ru
Web: <http://www.dmk.ru>



Книга содержит поэтапное описание и рекомендации по методике работы с программой цифрового видеомонтажа Adobe Premiere 5.0. Подробно рассмотрен пользовательский интерфейс и вопросы взаимодействия основного модуля с подключаемыми дополнениями, а также перечислены средства для работы с видео- и аудиоматериалами. Издание хорошо иллюстрировано, состоит из 14 глав и предметного указателя.

Это руководство предназначено всем, кто увлечен миром мультимедиа — и профессионалам, и начинающим, и любителям.

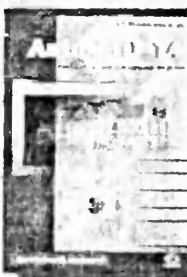


Книга знакомит с самой популярной версией языка Паскаль — Turbo Паскаль 7.0 фирмы Borland, а также его наиболее известным расширением — Борланд Паскаль 7.0. Содержит много примеров и алгоритмов. Новые понятия о структурах данных и средствах организации вычислительных процессов, а также их отладки, вводятся по мере возрастания сложности задач: от первой программы, которая умеет выводить на экран единственное сообщение, — до последней, протраивающей WAV-файлы; от обычных переменных — к динамическим объектам.



Книга предназначена для пользователей, желающих начать работу с ресурсами Internet в ближайшее время, но не знающих с чего начать. Подробно рассматриваются вопросы выбора и последующей настройки модема, приводятся критерии выбора провайдера. Приводится большое количество справочной информации необходимой при подключении к Internet, но не доступной вне сети.

В книге описываются все аспекты современного программирования на ассемблере для DOS, Windows 95/NT и Unix (Solaris, Linux и FreeBSD), включая создание резидентных программ и драйверов, прямое программирование периферийных устройств, управление защищенным режимом и многое другое. Подробно рассмотрена архитектура процессоров Intel вплоть до Pentium II. Все главы иллюстрированы подробными примерами работоспособных программ.



Книга знакомит с приложениями, входящими в состав программы FrontPage: проводником (FP Explorer), редактором (FP Editor) и персональным Web-сервером. В ней также описаны новые возможности последней версии программы FrontPage 98. Рассмотрены языки гипертекстовой разметки (HTML 3.2) и создания сценариев (JavaScript).



Книга является руководством по применению программного комплекса AutoCAD (версия 14 для Windows 95 и Windows NT). Данное издание поможет читателю в короткий срок освоить методы двумерного черчения, трехмерного моделирования и разработки конструкторской документации.

Навыки, полученные при изучении данной книги, могут быть с успехом использованы при работе с другими версиями программы AutoCAD.



Только пользующиеся спросом книги



Издательские цены, гибкая система скидок



Рекомендации по подбору ассортимента для каждого клиента



Склад в центре Москвы



264-75-36



books@dmk.ru



www.dmk.ru

Кудрявцев Евгений Михайлович

AutoLISP
Программирование в AutoCAD 14

Главный редактор Захаров И.М.
Научный редактор Сидорова Т.М.
Литературный редактор Готлиб О.В.
Верстка Волнов Ю.А.
Дизайн обложки Кудряшов А.В.

ЛР № 065625 от 15.01.98. Подписано в печать 15.06.1999.
Формат 70x100 $\frac{1}{16}$. Гарнитура «Петербург». Печать офсетная.

Усл. печ. л. 23. Тираж 3000. Заказ № 1278

Издательство «ЛАЙТ Лтд.», 113093, Москва, Б. Серпуховская, 8/7, стр. 2.

Отпечатано с готовых диапозитивов в ППП «Типография «Наука»
121099, Москва, Шубинский пер., 6.

В книге приводится развернутое описание новой мощной интегрированной среды программирования *Visual LISP* для системы *AutoCAD 14*. Рассматривается структура среды, ее запуск, интерфейс, текстовый редактор, отладчик и другие элементы *Visual LISP*.

Также рассмотрены базовые понятия и определения языка *AutoLISP*, основные этапы программирования на нем, встроенные функции языка и функции, расширяющие его возможности. Описывается программирование на языке *AutoLISP* и отладка программ в среде *Visual LISP*.

В доступной форме изложены процессы создания различных прототипов систем и подсистем с использованием языка *AutoLisp*: от параметрического представления изображений, выполнения расчетов различных объектов, проблемно-ориентированных систем обработки символьной и числовой информации, включая обработку статистических данных и систему управления базами данных, до экспертных систем и решения различных инженерных задач. Изложение сопровождается созданием соответствующего программного обеспечения на языке *AutoLISP*, которое может быть использовано при разработке аналогичных систем и подсистем.

Книга рассчитана на широкий круг читателей, желающих освоить и применять современные компьютерные технологии в своей практической деятельности.

ИНТЕРНЕТ-МАГАЗИН

WWW.BOOKS.RU



www.dmk.ru

ISBN 5-89818-023-0



9 785898 180232